View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Theseus

jamk.fi

KOULUTUSVIDEOIDEN JULKAISUJÄRJESTELMÄ

Markus Makkonen

Opinnäytetyö

Lokakuu 2013

Mediatekniikan koulutusohjelma

Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t)	Julkaisun laji	Päivämäärä
Markus Makkonen	Opinnäytetyö	30.09.2013
	Sivumäärä 45	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty (X)

Työn nimi

KOULUTUSVIDEOIDEN JULKAISUJÄRJESTELMÄ

Koulutusohjelma

Mediatekniikan koulutusohjelma

Työn ohjaaja(t)

Kari Niemi

Toimeksiantaja(t)

SKYNest-projekti, Jyväskylän Ammattikorkeakoulu, Teknologiayksikkö

Tiivistelmä

Opinnäytetyön toimeksiantaja oli SkyNEST-projekti. SkyNEST on Jyväskylän ammattikorkeakoulun Teknologiayksikössä ICT-tulosalueella toimiva projekti, joka on osa Tekesin rahoittamaan ICT-SHOKiin kuuluvaa Cloud Software-ohjelmaa. Opinnäytetyö toteutettiin SkyNESTin FreeNest-projektin käyttöön. FreeNest on tiimipohjainen tuotteenkehitysalusta.

FreeNestin käytön tueksi tarvitaan huomattava määrä erilaisia koulutusvideoita, joissa opastetaan jokainen toiminto, jonka tuotteella voi tehdä. Videoita tekevät usein FreeNestin kehittäjät, muun kehitystyön aikana. Ongelma on, että videoiden julkaiseminen tämän jälkeen on melko työläs prosessi. Videoihin täytyy editoida projektin omat aloitus- ja lopetussekvenssit, kuten alku- ja lopputekstit, sekä tämän jälkeen julkaista videot YouTubessa sopivalla nimellä ja selityksellä.

Opinnäytetyössä lähdettiin rakentamaan sovellusta, jolla tämä voitaisiin automatisoida. Kyseessä on web-pohjainen sovellus,jota käyttämällä videot voidaan rakentaa valmiiksi nopeasti ja tämän jälkeen automaattisesti renderöidä ne käyttäen palvelimen resursseja.

Opinnäytetyön tietoperusta käsittelee yleisiä web-teknologioita, kuten PHP, JavaScript yms, joita sovelluksen luonnissa käytettiin. Hieman tarkemmin paneudutaan Knockout.js framework:iin, joka helpottaa MVVM-mallin mukaisen sovellusarkkitehtuurin rakentamista.

Tuloksena saatiin sovellus, jolla pystyy esittelemään konseptin toimintaa. Sovelluksessa kaikki vaaditut asiat toimivat, mutta joitakin käyttöön vaadittavia lisäominaisuuksia ei saatu viimeisteltyä.

Avainsanat (asiasanat)

Broadcaster, PHP, JavaScript, Knockout.js

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Author(s)	Type of publication	Date
Markus Makkonen	Bachelor's Thesis	30.09.2013
	Pages 45	Language Finnish
		Permission for webpublication (X)

Title

Training Material Broadcaster

Degree Programme

Media Engineering

Tutor(s)

Kari Niemi

Assigned by

SKYNest project, JAMK University of Applied Sciences , School of Technology

Abstract

This Bachelor's Thesis was assigned by SkyNest-project. SkyNest operates at the School of Technology at JAMK University of Applied Sciences. SkyNest is a part of Cloud Software program of ICT-SHOK. This thesis was made to be used by FreeNest-project, which is a part of SkyNest. FreeNest is a team-oriented product development platform.

FreeNest needs a huge number of training videos to guide users through every function of the program. These training videos are often made by FreeNest-developers during their development work. Releasing these videos is quite laborious process, which may cause problems. Every video requires FreeNest's own ending and starting sequences. Every video also need to be released in YouTube with a suitable name and description.

The goal of this thesis was to create a program that could automate this process. This program is a web-application that can be used to build videos and automatically render and release them. Rendering uses the resources from the server-machine.

Theoretical part of this study discusses some of the common web-technologies, such as PHP and JavaScript. Thesis focuses on Knockout.js-framework more in detail, which is used to create MVVM-model based web-applications.

The result of this work was application that can be used to present the functionality of concept. All required functions work; howerver some of could not be finished within the scope of this thesis.

Keywords

Broadcaster, PHP, JavaScript, Knockout.js

Miscellaneous

SISÄLTÖ

KUVIOT

TERMIT JA LYHENTEET

Apache

Apache on avoimeen lähdekoodiin perustuva palvelinohjelmisto.

avconv

Komentorivipohjainen työkalu videotiedostojen muokkaamiseen.

Codec

Codec tai koodekki, on pakkausalgoritmi, joka muuntaa äänen tai videon vähemmän tallennus ja siirtokapasiteettia kuluttavaan muotoon.

Databindaus

Databindaus on tekniikka, jolla yhdistetään kaksi tietolähdettä toisiinsa automatisoiden näiden välisen synkronoinnin.

DOM-elementti

Document Object Model eli DOM sisältää tiedot siitä, miten HTML-sivu on muodostettu. DOM-elementti voi olla , <html>, <body> tai mikä tahansa muu HTML-elementti.

Framework

Kokoelma kirjastoja ja funktioita, joiden tarkoitus on tehdä tiettyjen toimintojen toteuttamisesta koodissa helpompaa.

Front-end

Sovelluksen käyttäjälle näkyvä osa.

HTML

HTML (Hypertext Markup Language) on web-sivujen rakenteen ja sisällön määrittelyyn käytetty kuvauskieli.

Library

Kirjasto on kokoelma erilaisia funktioita ja aliohjelmia, joita voidaan käyttää apuna ohjelmistojen kehittämisessä.

MP4Box

Komentorivipohjainen työkalu videotiedostojen muokkaamiseen.

Oliopohjainen

Ohjelmointimalli, joka perustuu erilaisia objekteja kuvaaviin luokkiin ja niistä luotuihin instansseihin.

Palvelin

Palvelin tarkoittaa tietokoneessa ajettavaa palvelinohjelmistoa, joka tarjoaa erilaisia palveluita muille ohjelmille, sekä tätä palvelinohjelmistoa ajavaa tietokonetta.

Skriptikieli

Skriptikieli tarkoittaa kieltä, jossa koodia ei käännetä konekielelle ennen suorittamista, vaan koodi suoritetaan sellaisenaan koodimoottorin avulla.

Tietokanta

Tietokanta tarkoittaa tietovarastoa, johon kootaan tietoja, joilla on yhteyksiä toisiinsa.

UI

UI (User Interface) eli käyttöliittymä

1. LÄHTÖKOHDAT

1.1 Toimeksiantaja

Työn toimeksiantaja oli SkyNEST-projekti. SkyNEST on Jyväskylän ammattikorkeakoulun (JAMK) Teknologiayksikössä ICT-tulosalueella toimiva projekti.
SkyNEST on osa Tekesin rahoittamaan ICT-SHOKiin kuuluvaa Cloud Software ohjelmaa (http://www.cloudsoftwareprogram.org/). Cloud Software-ohjelman tarkoitus on kehittää ratkaisuja, joilla voitaisiin parantaa suomalaisen ohjelmistoteollisuuden kilpailukykyä. (Projektit, 2013)

Opinnäytetyö tehtiin osana SkyNESTin kesätehdasta, joka on noin 30 vaihtuvan opiskelijan ryhmä, joka kehittää työharjoitteluna SkyNESTin projekteja JAMK:issa. Valmista opinnäytetyö on tarkoitus tulla käyttämään SkyNESTin FreeNest-projektissa. FreeNest on tiimikeskeinen tuotteen kehitysalusta, jonka ideana on yhdistää muut, tuotteen kehityksessä yleisesti käytössä olevat open source sovellukset yhden ohjelmiston alle. (Freenest-Product Platform, 2013)

1.2 Tehtävä ja tavoitteet

Opinnäytetyön tavoitteena oli kehittää koulutusvideoiden julkaisujärjestelmä. Valmista työtä tultaisiin käyttämään FreeNest-projektin koulutusvideoiden julkaisussa. Valmis työ voidaan julkaista myös GitHubissa. GitHub on web-pohjainen ylläpitopalvelu ohjelmistoprojekteille, jotka käyttävät Git-revision hallintaohjelmistoa.

FreeNest tarvitsee huomattavan määrän erilaisia koulutusvideoita opettamaan ohjelman jokaisen toiminnon. Videoita tekevät monesti muun kehitystyön aikana FreeNestin kehittäjät. Ongelma on, että videoiden julkaiseminen tämän jälkeen on melko työläs prosessi. Videoihin täytyy editoida projektin omat aloitus- ja lopetussekvenssit sekä tämän jälkeen julkaista video YouTubessa sopivalla nimellä ja selityksellä. Pelkän videokoosteiden rakennuksen lisäksi sovellukselta vaadittiin myös joitakin lisäominaisuuksia.

Sovelluksen täytyy kyetä tekemään siirtymävideoita käytettäväksi osien välissä pitkissä videoissa. Alunperin oli ideana, että sovellukseen voisi kopioida jonkin tekstin pätkän, josta sovellus sitten loisi videon, mutta tämä todettiin hankalaksi toteuttaa. Sovelluksessa päädyttiin ratkaisuun, että siirtymävideot tehdään kuvatiedostojen pohjalta. Kuvat todettiin monikäyttöisemmiksi ja ratkaisu helpommaksi toteuttaa, (internetistä löytyy suoraan työkaluja jotka rendaavat videon kuvatiedostosta). Siirtymien tekijä voi käyttää mitä tahansa ohjelmaa luomaan kuvia, joissa teksti on suoraan sopivalla fontilla ja rivityksellä. Näiden toimintojen lisääminen sovellukseen itseensä olisi voinut olla hankalaa.

Sovelluksen pitäisi myös kyetä peilaamaan muutoksia useampaan videoon. Esimerkiksi jos käyttäjä haluaa vaihtaa videoiden aloituslogon, voisi käyttäjä valita yhden videon, joka sisältää tämän aloituslogon. Tämän jälkeen käyttäjä pystyisi valitsemaan korvaavan videon ja sovellus tekisi automaattisen korvauksen kaikkiin videoihin, jossa tämä osa esiintyy. Tämän jälkeen sovellus rendaisi kaikki muuttuneet videot uusiksi ja lähettäisi niistä päivitykset Youtubeen. Tämä kaikki tapahtuisi automaattisesti palvelimella ja käyttäjä voisi seurata toimenpiteen etenemistä front-endistä käsin.

1.3 Tekniset ratkaisut

Työstä oli tarkoitus tulla palvelinsovellus, joka pyörisi apache2-palvelimen päällä ubuntu-ympäristössä. Sovellus olisi käytettävissä web-käyttöliittymän kautta. Käännöksiä muihin ympäristöihin, tai mahdollisuuksia käyttää sovellusta muilla laitteilla ei suunniteltu tämän projektin yhteydessä. Web-käyttöliittymä suunniteltiin niin, että se olisi helposti käytettävissä normaalilla web-selaimella työpöytäkoneella. Käyttöliittymän sovittaminen esimerkiksi mobiililaitteille ei kuulunut tähän projektiin.

Web-käyttöliittymän front-end on yksi HTML-sivu. Sivun toteutus tehtiin JavaScriptillä käyttäen Knockout.js-frameworkia, joka helpottaa huomattavasti responsiivisen web-ui:n rakentamista.

Sovelluksen täytyy kyetä hyödyntämään palvelinkoneen resursseja videoiden rendaamisesta ja rakentamisesta, joten tarvittiin myös osia, jotka pyörivät palvelimella, ja web-käyttöliittymä täytyi saada toimimaan saumattomassa yhteistyössä palvelimella olevien osien kanssa. Palvelinpuolen osien toteuttamisessa päädyttiin käyttämään Python-skriptausta, jonka kanssa web-sovellus keskustelisi PHP-tiedostojen välityksellä käyttäen Ajax-kutsuja.

1.4 Käytetyt ohjelmat

Kaikki sovelluksen PHP, JavaScript ja HTML on kirjoitettu käyttäen PHPStorm-kehitysympäristöä. PHPStorm on Jetbrainsin (http://www.jetbrains.com/phpstorm/) kehittämä ohjelmointiympäristö PHP:n, JavaScriptin ja HTML:n työstämiseen. PHPStorm valittiin pääasialliseksi työkaluksi opinnäytetyön toteuttamiseen, koska kyseisestä sovelluksesta oli jo aikaisempaa kokemusta ja se sisältää hyvin kattavat työkalut PHP:n ja JavaScriptin tekoon sekä testaukseen.

Muita käytettyjä ohjelmia oli Sublime Text (http://www.sublimetext.com/), Pythonin kirjoitukseen ja Chrome-selain testaukseen sekä virheiden etsintään. Chrome sisältää erinomaisen Inspect-työkalun, joka osaa näyttää virheet kattavasti JavaScript-koodissa.

Sovellus pyörii Apache2-palvelinohjelmiston päällä Ubuntussa. Ubuntua käytettiin myös käyttöjärjestelmänä työkoneella, jolla opinnäytetyö tehtiin, jolloin kaikkien osien toimivuutta voitiin testata suoraan. Myöskin joissakin sovelluksen osissa tarvittujen komentorivityökalujen toimivuus päästiin näkemään suoraan.

2. KOULUTUSMATERIAALIN JULKAISUJÄRJESTELMÄT

2.1 Selvitys vastaavista olemassa olevista järjestelmistä

2.1.1 Yleistä

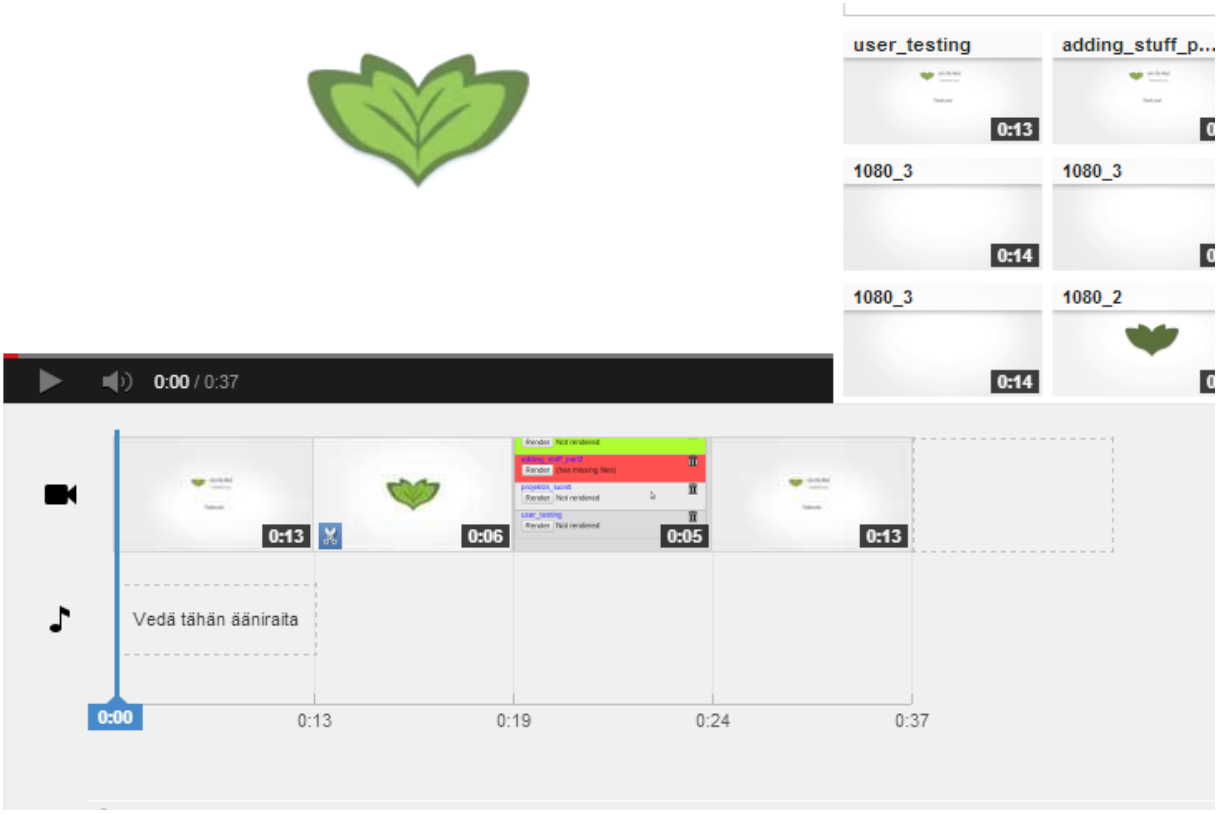
Työssä lähdettiin tutustumaan jo olemassa oleviin web-käyttöisiin videon editointisovelluksiin. Näitä on olemassa itse asiassa yllättävän monia. Monet näistä ovat hyvinkin monipuolisia ja helppokäyttöisiä, yleensä näissä on ongelmana lähinnä rajallinen videoiden tallennuskapasiteetti. Tämä rajoittaa paljon videoiden maksimiresoluutiota ja kestoa. Joissakin palveluissa nämä ovat myös suoraan rajattuja. Osassa nämä pystyy kiertämään käyttämällä kolmannen osapuolen varastoja videoille. Esimerkiksi Pixorial (http://www.pixorial.com/) tukee Google Driveä, joka antaa 5 GB:tä tallennustilaa. Suoraan tämän työn vaatimuksia täyttävää

editoria ei löytynyt, mutta joitakin joista suurin osa ominaisuuksista löytyy.

2.1.2 YouTuben editori

YouTube sisältää oman videoeditorin (http://www.youtube.com/editor), jolla on mahdollista yhdistellä omalle YouTube-tilillesi ladattuja videoita. Tämä tosin vaatii, että kaikki videon osat on lähetetty YouTubeen erikseen, muuten ne eivät ole tämän editorin muokattavissa. YouTuben editorin etuina on loputon tallennustila. Tämä editori on myös helppokäyttöinen ja mahdollistaa kaikki oleelliset toiminnot, kuten videoiden paloittelun ja yhdistämisen (ks. kuvio 1). YouTubella on myös maailman laajin kirjasto (yli 4 miljoonaa) Creative Commons-lisenssin alaisia videoita, joita voi käyttää osina omia videokoosteita. Creative Commons on lisenssi, jonka alaista materiaalia saa käyttää vapaasti johdannaisteoksissa, sillä ehdolla, että alkuperäisen tekijän nimi mainitaan. On myös CC0 lisenssi, jossa tekijän nimeä ei tarvitse mainita, mutta tämä ei ole mahdollista Suomessa. (Tietoja Lisensseistä)

YouTuben editori sisältää myös hyvän määrän työkaluja erilaiseen videoiden muokkaukseen. Esimerkiksi videoon on mahdollista asettaa monenlaisia värifilttereitä ja muita tehosteita (ks. kuvio 2). YouTuben editorilla on myös mahdollista näyttää kuvatiedostoja videoissa. Ainoa puuttuva ominaisuus, on mahdollisuus korvata sama videon osa useammassa eri videosta yhtä aikaa. Tästä huolimatta nopean tutustumisen jälkeen YouTuben sisäinen editori on ehkä lähimpänä sitä mitä tältä työltä haettiin. YouTuben editorissa on paljon haluttuja toimintoja. Siinä ei myöskään ole rajoituksia tallennustilan tai videon resoluution suhteen. Käyttöliittymä on myös erinomaisen selkeä.



KUVIO 1. YouTuben videoeditorin käyttöliittymä.

KUVIO 2. YouTuben videoeditorin suodattimia.

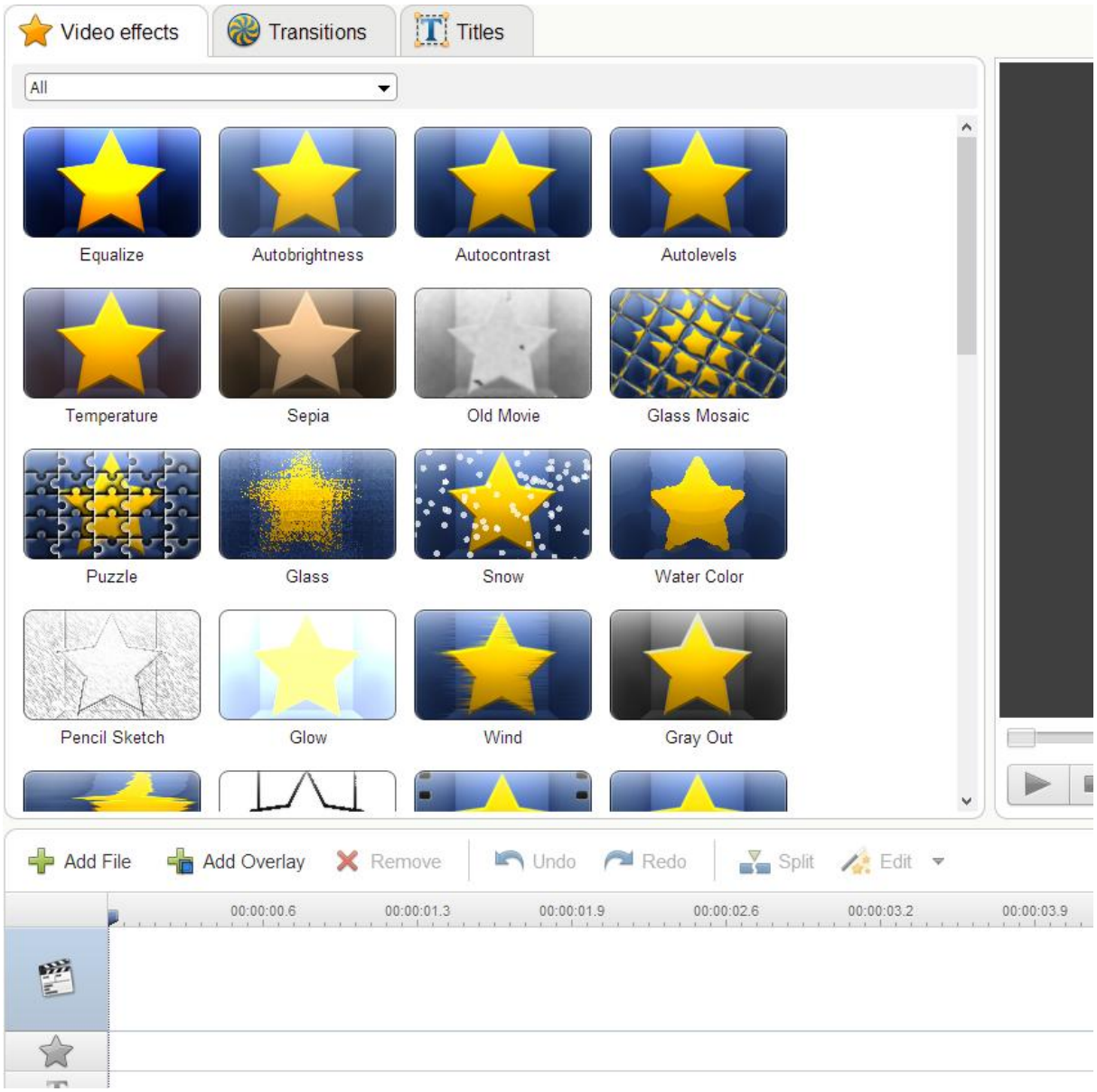
2.1.3 Muita web-pohjaisia videoeditoreita

Muita vastaavanlaisen web-editoreita on myös Pixorial (http://www.pixorial.com/) ja FileLab (https://videoeditor.filelab.com/app/).

Pixorial sisältää normaalin videoiden muokkauksen lisäksi mahdollisuuden nauhoittaa suoraan videota koneeseen liitetyllä web-kameralla. Pixorial sisältää myös omaa materiaalia lisättäväksi videoihin, mutta paljon suppeammin kuin YouTuben editori. Pixorial sisältää myös huomattavan varastotilan johtuen tämän tuesta Google Driveen.

FileLabin editori sisältää todella kätevän käyttöliittymän (ks. kuvio 3). FileLabin editori ei sisällä omia arkistovideoita lisättäväksi videokoosteisiin, mutta siinä on huomattava määrä tehosteita ja erilaisia siirtymiä. FileLab tukee myös YouTuben tapaan kuvatiedostojen asettamista aikalinjalle, jolloin niistä luodaan automaattisesti videoita.

Lisää web-pohjaisia videoeditoreita on listattu myös makeuseof.com:in erinomaisessa artikkelissa: 5 Free Tools For Online Video Editing. (http://www.makeuseof.com/tag/5-free-tools-online-video-editing/)



KUVIO 3. FileLabin videoeditorin käyttöliittymä.

2.1.4 Joitakin päätelmiä web-pohjaisista videoeditoreista.

Web-pohjaiset editorit eivät vastaa "oikeita" videoeditoreita toiminnallisuuden määrältään ja tehokkuudeltaan, mutta niillä on silti monia etuja. Codeceista ei tarvitse huolehtia ollenkaan, vaan ne ovat yleensä suoraan sivuston puolelta kunnossa. Ne ovat myös monesti "helppo" ratkaisu videoeditointiin, ohjelmat on suunniteltu hyvin helpoiksi käyttää ja kaikki tekninen puoli, esimerkiksi aiemmin mainitut codecit, on automatisoitu. Tietysti tämä pienentää samalla käyttäjän vaikutusmahdollisuuksia lopullisesta videosta.

2.2 Massive Open Online Courses (MOOC)

Mooc tarkoittaa tapaa järjestää verkkokursseja, joissa voi olla hyvin suuri määrä osallistujia, aina sadoista moniin tuhansiin. Moocin perusajatus on, että kurssille osallistumisen kynnys on mahdollisimman matala ja kurssin voi jättää milloin vain kesken ilman seuraamuksia. (mooc.cs.helsinki.fi)

Varsinainen opetus tapahtuu yleensä videostreamien välityksellä. Tämän lisäksi Mooceissa on opetuksen tukena muunmuassa kurssin omat keskustelupalstat ja irc-kanavat, joilla kurssin opettajat ja muut vapaaehtoiset neuvovat kurssille osallistuneita opiskelijoita. Hyvin laajoissa Mooceissa, joissa on tuhansia osallistujia, neuvonta perustuukin pitkälti opiskelijoiden keskinäiseen apuun, joiden tukena edellä mainitut kanavat ovat.

Syksyllä 2011 Sebastian Thrun järjesti Stanfordin yliopistossa tekoälykurssin, jolle osallistui yli 160 tuhatta opiskelijaa. Tämän jälkeen Moocit ovat olleet kuuma puheenaihe yliopisto-opetuksessa. Useat huippuyliopistot, kuten Stanford, Harvard ja MIT ovat lähteneet järjestämään useita Mooceja.

Moocit ja muutverkkokurssit ovat muuttamassa yliopisto-opiskelua. Tietojenkäsittelyn luennoilla käy entistä vähemmän väkeä, vaikka opettajat panostavat niihin entistä enemmän. (Vairimaa, R, 2013)

3. KÄYTETYT TEKNIIKAT

3.1 PHP

PHP on serveripohjainen Skriptikieli. Ensimmäisen version PHP:stä kehitti Rasmus Lerdorf vuonna 1994. (History of PHP, 2013) PHP:n syntaksi on suurimmaksi osaksi lainattu C-kielestä, Javasta ja Perlistä. Tähän on lisätty joitakin puhtaasti PHP-kohtaisia ominaisuuksia. (General Information, 2013) PHP on nykyään maailman ylivoimaisesti käytetyin serveripuolen kieli.

PHP on pääosin keskitetty serveripuolen skriptaukseen. PHP:n suoritus tapahtuu myös serverillä eikä selaimessa. PHP:n ajamiseen tarvitsee siis toimivan web-serverin, jossa on PHP asennettuna. PHP:llä voi tehdä myös komentoriviskriptausta. PHP:ta on mahdollista käyttää

myös työpöytäsovellusten luonnissa. Tosin PHP:tä ei ole varsinaisesti tarkoitettu tähän, ja moni muu työpöytäsovelluksiin suoraan suunnattu kieli on useimmiten paljon parempi valinta.

3.2 JavaScript

JavaScript on alunperin Netscapen kehittämä, selaimessa suoritettava olio-pohjainen (Object-oriented) skriptikieli. (MikaBug, 2007) JavaScript koodin suorittaa useimmiten selaimen JavaScript-moottori (Javascript-engine). JavaScriptiä ei pidä sekoittaa Javaan, sillä niillä ei ole mitään tekemistä keskenään. Nykyään JavaScriptiä käytetään lähes kaikilla web-sivustoilla ja se onkin maailman yleisimmin käytetty ohjelmointikieli. (JavaScript introduction, 2013)

JavaScript mahdollistaa monipuolisen kommunikoinnin käyttäjän kanssa. JavaScript voi esimerkiksi lukea sivuston lomakkeita ja antaa käyttäjällä palautetta niiden pohjalta. JavaScript myös mahdollistaa monipuolisen DOM-elementtien muokkauksen, jolloin sivustoa pystyy muokkaamaan lähes vapaasti. JavaScript on usein välttämätön osa minkä tahansa web-sovelluksen luomista.

JavaScript voidaan upottaa suoraan HTML-sivulla <script>-tagien sisässä tai se voidaan liittää siihen erillisestä tiedostosta.

3.3 Ajax

Ajax tulee sanoista Asynchronous JavaScript And XML. Ajax on yleisnimitys joukolle web-kehitystekniikoita, joilla web-sovelluksista pyritään tekemään vuorovaikutteisempia. (Garrett, J, 2005) Klassisessa web-kehitysmallissa tyypillisesti ladataan koko sivu uusiksi tarvittaessa hakea jotakin uutta tietoa palvelimelta. Ajax-tekniikoissa ideana on, että sen sijaan että ladataan koko sivu uudestaan pyydettäessä uutta tietoa, pyyntö menee Ajax-käsittelijään, joka lähettää palvelimelle pyynnön tiedosta ja palvelin lähettää vain tämän tiedon. Tämän jälkeen Ajax-käsittelijä lukee tiedon ja tarjoaa sen web-sivuston skriptille, yleensä JavaScript, joka muokkaa sivuston front-endin vastaamaan uutta tietoa.

3.4 JQuery

JQuery on vuonna 2006 julkaistu avoimeen lähdekoodiin perustuva, ilmainen JavaScript-kirjasto. (JQuery Foundation, 2013) JQuery on kehitetty yksinkertaistamaan JavaScriptin syntaksia tarjoamalla muun muassa helppokäyttöisiä tapoja käsitellä sivuston DOM-elementtejä, tapahtumia ja Ajax-kutsuja. JQueryä käyttävät nykyään miljoonat web-sivustot ja se maailman suosituin JavaScript-kirjasto.

3.5 Event-Driven Programming

Event-Driven Programming on ohjelmointityyli, jossa sovelluksen eteneminen on säädelty tapahtumien (events) pohjalta. Näitä ovat esimerkiksi, käyttäjän klikkaukset, viestit toisilta ohjelmilta, syötteet erilaisilta sensoreilta yms.

Event-Driven Programming-sovellus toimii tavallisesti niin, että se sisältää vain suuren määrän erilaisia tapahtumankuuntelijoita, jotka tekevät toimenpiteitä kun käyttäjä tekee jotain. Eroten tässä perinteisestä Procedural Programming-metodista, jossa sovelluksessa on yleensä yksi jatkuvasti ajettava päälooppi, joka tekee toimenpiteitä riippumatta käyttäjästä ja seuraa tapahtumia koodin suorituksen lomassa. Event-Driven Programming-ohjelmointityyli taas ajaa koodia tapahtumien pohjalta.

Event-Driven voidaan toteuttaa myös sisältäen Procedural Programming-tyylin tapaan pääloopin, jossa ohjelmaa suoritetaan. Tällöin päälooppi on vain jaettu kahteen erilliseen osioon, jossa ensimmäinen seuraa tapahtumia ja toinen toteuttaa vastaukset toteutuneisiin tapahtumiin.

Event-Driven Programming on usein järkevin ohjelmointitapa UI:n käsittelyyn, jossa vastaukset käyttäjän toimenpiteisiin ovat keskeisin asia, jota ohjelma tekee.

3.6 MVC-arkkitehtuuri

MVC (Model-View-Controller) on ohjelmointiarkkitehtuuri, jossa ideana on erottaa toisistaan, ohjelman tietovarasto ja tiedon käsittelijä (Model), ohjelman käyttöliittymä (View), käyttäjän vuorovaikutus käyttöliittymän kanssa (Controller). MVC kehitettiin alunperin Smalltalk-ohjelmointikielen yhteydessä hyvin varhaisten graafisten käyttöliittymien luonnin helpottamiseksi 80-luvulla. (Davis, 2008) Mutta on sittemmin havaittu erittäin käytännölliseksi webin kautta käytettävien sovellusten rakentamisessa.

MVC:ssä Model on ohjelman sydän, joka vastaa kaikesta ohjelman tiedosta. Tämä sisältää ohjelman muuttujat. Model vastaa myös kaikesta tiedon jakamisesta muihin osiin sekä tiedon lisäämisestä ohjelman muistiin. Tieto voi tulla joka Controllerista käyttäjän syöttämänä tai se voidaan hakea ulkopuolisesta lähteestä. Ilman Modelia ei ole ohjelmaa, sillä kaikki tieto Viewin ja Controllerin välillä kulkee Modelin kautta.

View on paikka, jossa kaikki Modelilta pyydetty tieto esitetään käyttäjälle. Yleensä web-sovelluksissa View on sovelluksen osa, joka generoi käyttäjälle näkyvän HTML-rakenteen. View myös aloittaa käyttäjältä tulevat reaktiot. Esimerkkinä Viewin luoma nappi, jota käyttäjä painelee ja käynnistää toiminnon Controllerissa.

Controller on osa, joka käsittelee kaikki käyttäjältä tulevat toiminnot. Controlleria ei olisi olemassa ilman käyttäjää; kaikki mitä Controller tekee, on jotakin, mitä käyttäjä laittoi sen tekemään. Controller ei sisällä mitään muuta ohjelmointilogiikkaa kuin sen mitä tarvitaan lukemaan käyttäjältä tuleva syöte. Controller lähettää kaiken tiedon Modelilla, jossa tieto sitten prosessoidaan ja tallennetaan ohjelman tietovarastoon.

3.7 MVVM-arkkitehtuuri

MVVM eli Model-View-View Model on ohjelmointi-arkkitehtuuri joka on kehitetty erityisesti tapahtumapohjaisten ohjelmointialustojen (kuten HTML5) käyttöön. MVVM on kehitetty osittain MVC:n pohjalta. MVVM-arkkitehtuuri koostuu kolmesta osasta, jotka eroavat MVC:n osista. MVVM:n osat ovat ohjelman tietovarasto ja käsittelijä (Model), koodimuotoinen esitys

ohjelman sisältämästä tiedosta (View-Model) ja käyttäjälle näkyvä ohjelman osa (View) (Implementing the MVVM pattern, 2013)

MVVM:ssä View Model sisältää esityslogiikan ja tarvittavan datan Viewiä varten. View Modelilla ei ole mitään suoraa tietoa siitä, kuinka View tulee esittämään datan. View Model ainoastaan tarjoaa määritykset ja komennot, joita Viewiin voidaan databindata. View Model myös ilmoittaa Viewille aina kun ohjelman sisältämä data muuttuu, jotta View voi sopeutua muutoksiin. View Model on yhteydessä moniin Modeleihin, joista tietoa haetaan tarpeen mukaan.

View on vastuussa siitä, mitä käyttäjälle esitetään ja minkälaisessa muodossa. Ainoa koodi, jonka View sisältää, on UI-logiikka, joka määrittelee, miten UI rakennetaan perustuen View Modelin dataan.

Model on tietovarasto joka yhdistää View Modeleita. Modeliin tallennetaan tieto jonka täytyy olla saatavilla monessa osassa sovellusta. Model ei sisällä esityslogiikkaa tiedosta.

3.8 Knockout.js

3.8.1 Taustaa

UI:n rakentaminen dynaamisesti voi olla hyvin työlästä ilman tähän soveltuvaa frameworkkia. Tätä helpottaa huomattavasti MVVM:n mallin mukainen kaksisuuntainen databindaus. Tätä ei voi oikeastaan toteuttaa normaalilla Javascriptilla, joten tehtävään tarvittiin framework. Tähän tehtävään valittiin Knockout.js.

Knockout.js on alunperin vuonna 2010 julkaistu Microsoftin työntekijän Steve Sandersonin kehittämä open-source JavaScript-framework. Ensimmäinen stabiili versio Knockoutista julkaistiin 2013. Knockout.js tukee kaikkia keskeisiä selaimia ja on erittäin kompakti. (Knockout: Introduction, 2013)

Knockout.js sisältää erittäin näppäriä pitkälle automatisoituja toimintoja, jotka liittyvät datan muokkaamiseen vaadittavien työkalujen toteuttamiseen UI:ssä. Varjopuolena Knockout.js-sovellus muuttuu hankalaksi ylläpitää sovelluksen kasvaessa, mutta tämä sovellus ei ole

äärimmäisen laaja, joten knockout.js katsottiin sopivaksi valinnaksi.

3.8.2 Parannus JQueryyn

Knockout.js ei ole suunniteltu korvaamaan JQueryä, vaan toimimaan sen kanssa. Kun UI alkaa muuttua laajaksi ja sisältää paljon keskenään ristiin toimivaa käytöstä, alkaa sen rakentaminen pelkällä JQueryllä olla hyvin työlästä. Esimerkiksi jos listaan täytyy tehdä nappi, joka poistaa siitä dataa, tämä voi olla hankalaa perus-Jqueryllä, pitää muun muassa selvittää, mikä elementti on kyseessä. Kun taas Knockout voi tehdä tämän hyvin yksinkertaiseksi huolehtimalla datan ja UI-elementtien yhteyksistä automaattisesti.

3.8.3 Knockout.js:n View Modelit

Knockoutin toiminta perustuu View Modeleihin. View Modelin luonti Knockoutissa on helppoa. Hyvin yksinkertaisen viewmodelin luonti JavaScriptillä:

```
var myViewModel = {
        videoName: 'Adding Items',
        videoLength: 12
};
```

View Model täytyy myös muistaa ottaa käyttöön.

```
ko.applyBindings(myViewModel);
```

Oletuksena View Model on käytössä sivuston koko HTML-rakenteessa, mutta se voidaan ottaa käyttöön vain osaan sitä, jolloin tehostetaan koodin suorittamista. Tämä tehdään spesifioimalla haluttu DOM-elementti ottaessa View Modelin käyttöön.

```
ko.applyBindings(
        myViewModel,
        document.getElementById("videoContainer")
);
```

Kun View Model on kunnossa, täytyy laittaa View-puoli kuntoon HTML:ssä. Hyvin yksinkertaisen Viewin voi tehdä helposti käyttäen selittävää databindausta. Alla oleva esimerkki luo <Span> tagin jonka sisältö määräytyy View Modelin videoName-muuttujen perusteella:

```
Video is <span data-bind="text: videoName"></span>
```

Tästä tulostaa sivulle seuraava ilmoitus.

```
Current video is Adding Items!
```

3.8.4 Observablet

Knockoutin suurimpina etuja on automaattinen UI:n päivittäminen, kun View Model muuttuu. Tämä tehdään käyttäen Observableja. Observablet kertovat knockoutilla, minkä muuttujien kuuluu päivittyä UI:seen automaattisesti. Esimerkkinä aiempi View Model observable bindauksilla:

```
var myViewModel = {
        videoName: ko.observable('Adding Items'),
        videoLength: ko.observable(12)
};
```

Nyt jos koodissa tekee seuraavan muokkauksen:

```
myViewModel.videoName('Deleting Items');
```

HTML-sivu muuttuu automaattiseksi seuraavanlaiseksi:

```
Current video is Deleting Items!
```

Observableja muokatessa täytyy ottaa huomioon, että jokainen observable on itse asiassa funktio, joten pelkkä videoName = "Deleting Videos" on väärin. Muutos täytyy tehdä kuten funktiolle syöttäisi eli sulkeiden sisässä. Myöskin observableja lukiessa on muistettava sulkeet. videoName(). Viewissä eli HTML-sivulla sulkeita ei tarvitse.

3.8.5 Muita hyödyllisimpiä toimintoja

Foreach-bindaus on tarkoitettu luomaan graafisia esityksiä listoista. Esimerkiksi seuraavanlainen bindaus HTML:ssä luo taulun, joka listaa kaikki eri videos-arrayn alla olevat videot allekkain näkyviin web-sivulle.

```
<table>
   <tbody data-bind="foreach: videos">
      <tr>
         <td data-bind="text: videoName"></td>
         <td data-bind="text: videoLength"></td>
      </tr>
   </tbody>
</table>
```

3.9 Muut kirjastot

Laajensin Knockoutin toiminnallisuutta joillakin kirjastoilla. Knockout-sortable (https://github.com/rniemeyer/knockout-sortable) lisää erittäin käytännöllisen uuden bindaustavan knockoutiin. Käytännössä uusi sortable-bindaus on kuin foreach, mutta se sisältää suoraan toiminnallisuuden listan sorttaamiseen manuaalisesti.

Esimerkiksi seuraava koodi luo luettelon videoista. Tätä luetteloa voi järjestellä uudestaan raahaamalla sen osia hiirellä eri järjestykseen.

```
<div data-bind="sortable: videos">
      <div>
            <div style=”float:left”
            data-bind="text: videoName"></td>
            <div style=”float:left”
            data-bind="text: videoLength"></td>
      </div>
</div>
```

3.10 MySQL

Kaikki sovelluksessa tarvittava data tallennetaan tietokantaan, joka tässä sovelluksessa on palvelimella ajettava MySQL-kanta.

MySQL on maailman suosituin tietokantaohjelmisto, jota on ladattu yli 100 miljoonaa kertaa. (about MySQL, 2013) MySQL perustuu avoimeen lähdekoodiin. MySQL:n kehitti alunperin Michael ”Monty” Widenius ja hänen kollegansa vuonna 1994. Tuotteen julkisti TcX Dataconsult AB, joka myöhemmin vaihtoi nimensä MySQL AB:ksi. (FAQ on MySQL vs. NuSphere Dispute, 2001) Monet maailman suurimmista organisaatioista käyttävät MySQL:ää.

3.11 Python

Python on alunperin 1980-luvun loppupuolella kehitetty ohjelmointikieli (General Python FAQ, 2013) ja kyllä, Python on nimetty Monty Pythonin lentävän sirkuksen mukaan.

Python on huomattavan tehokas serveripuolen kieli, joka sisältää erittäin selkeän syntaksin. Pythonin syntaksi tekee koodista myös erittäin lyhyttä, usein monta kertaa lyhyempää kuin C tai Java. (Why use Python, 1999) Pythonia voi ohjelmoida sekä oliopohjaisella että proseduraalisella ohjelmointimallilla. Kieli on suunniteltukin mahdollisimman joustavaksi ja syntaksiltaan pelkistetyksi.

3.12 Video-pluginit

3.12.1 Videoiden rakennus

Sovelluksen täytyy kyetä yhdistämään ainakin MP4-formaatissa olevia videoita. Tuki myös OGV-videoille oli suunnitteilla, mutta tätä ei ehditty toteuttaa tämän projektin sisällä. Näin ollen tarvittiin ohjelma, joka osaa yhdistää peräkkäin vähintään MP4-formaatissa olevia videoita. Ohjelman täytyi myös osata skaalata molemmat videot samalle resoluutiolle ja frameratelle.

Tähän päädyttiin käyttämään MP4Box-työkalua. MP4Box on komentorivipohjainen työkalu, joka asentuu suoraan palvelimelle ja on täten web-sovelluksen käytettävissä palvelinpuolen koodin kautta. MP4Box sisältää monipuoliset työkalut kaikkien oleellisten videoformaattien muokkaukseen, esimerkiksi MP4, AVI, MPG ja 3GP (MP4Box | GPAC, 2013). MP4Box on myös huomattavan nopea.

Alla esimerkki komennosta, jolla MP4Box yhdistää videot alku.mp4 ja loppu.mp4 yhdeksi pitkäksi videoksi ja tallentaa tämän nimellä full.mp4 kansioon Completed. Logi toiminpiteestä tallennetaan tiedostoon log.txt.

```
MP4Box -cat alku.mp4 -cat loppu.mp4 /Completed/full.mp4 > log.txt
```

3.12.2 Siirtymävideoiden rendaaminen kuvatiedostoista

Sovellukseen tarvittiin työkalua, joka voisi ottaa kuvatiedoston ja tehdä siitä halutun mittaisen videon, jossa kuvaa vain esitetään ruudulla haluttu aika. Ohjelman täytyi myös osata tehdä video halutulla resoluutiolla, joka voisi olla eri kuin kuvatiedoston resoluutio.

Tässä päädyttiin käyttämään avconv-työkalua. Avconv on komentorivipohjainen videonmuokkaustyökalu, joka sisältää tähän tehtävään tarpeellisen toiminnon, dia-show tyyppisen videon rakentamisen kuvatiedostoista. Dia-shown voi luoda käyttäen vain yhtä kuvaa, jolloin tuloksena on juuri tähän tarkoitukseen sopiva välivideo.

Esimerkki komennosta, jolla luodaan viiden sekunnin mittainen video, kuvasta intermission.png. Videon resoluutioksi määritellään 640x480

```
avconv -loop 1 -f image2 -i intermission.png -t 5 -s 640x480
```

3.13 Youtube-pluginit

Sovellukseen tarvittiin jokin työkalu, jolla videoiden lataamisen Youtubeen voisi automatisoida. GoogleCL sisältää kaikki oleelliset toiminnot videoiden käsittelyyn.

GoogleCL eli Google Command Line on Googlen kehittämä API, jolla voi käyttää useimpia Google-accounttiin liitettyjä palveluita komentolinjalta käsin. GoogleCL asentuu suoraan palvelimelle ja on täten web-sovelluksen käytettävissä palvelinpuolen koodin kautta.

Tehtävässä testattiin myös Youtube-uploaderia (http://code.google.com/p/youtube-upload/), mutta päädyttiin käyttämään GoogleCL:n paremman tietoturvan ja helpomman käytettävyyden

vuoksi. GoogleCL käyttää OAuth-metodia tunnistaumiseen, joten salasanoja ei tarvitse olla missään paljaana tekstinä, kuten Youtube-uploaderissa.

Seuraavanlainen esimerkkikomento lähettää testiaccountille videon deleteItems.mp4 ja antaa sille kuvauksen. Kategorian määrittely on GoogleCL:ssä pakollista, käytin kaikkiin videoihin kategoriaa ”Education”

```
Google youtube post –category Education –summary ”How to delete items” , --user testailuaccountti deleteItems.mp4
```

4. SOVELLUKSEN TOTEUTUS

4.1 Layout

Layout eli sivun asettelu on pohjakoodi, jonka mukaan selain esittää sivuston. Tähän sovellukseen riitti, että se on helposti käytettävissä normaalilla selaimella ja normaalilla näyttöresoluutiolla (1920x1080), joten sovelluksen koko päätettiin tältä pohjalta. Layoutin skaalautuminen muille resoluutioille ei ollut oleellista. Myöskään sovelluksen toimiminen eri selaimilla ei ollut tärkeää. Sovellus toteutettiin niin, että se toimii hyvin kehityksessä käytetyllä Chrome-selaimella. Sovellus saattaa toimia myös muilla selaimilla, mutta kunnollisia testauksia ei ole tehty. Sivuston rakenteen hajoaminen on yksi uhka, mutta on hyvin mahdollista, että kaikki sovelluksessa käytetyt JavaScript-komennot eivät ole suoraan yhteensopivia muille selaimille.

Pohjaa sovellukselle lähdettiin rakentamaan videoiden muokkaus-osion pohjalta. Alkuperäinen ajatus oli putkimalli, jossa valittaisiin videon osia pala kerrallaan (alku, video, loppu) ja siirryttäisiin aina seuraavalla sivulle, jossa valitaan seuraava osa. Mutta tämä todettiin turhan monivaiheiseksi ja työlääksi käyttää. Lisäksi tämä olisi ollut ongelmallinen jos varsinainen video halutaan koostaa monesta eri palasta.

Käyttöliittymä johon päädyttiin on jaettu kolmeen sarakkeeseen. Tiedostolistaus, muokattavana oleva videokooste ja lista tallennetuista videokoosteista. Tämä pyrittiin pitämään sovelluksen pohjarakenteena kaikissa eri osioissa. Nämä sarakkeet olisivat aina samoja mahdollisuuksien mukaan eri osioissa. Sovelluksen varsinaisen graafisen layoutin suunnitteli FreeNestin

kesätehtaan designteam.

4.2 HTML-sivun toteutus

Sovelluksessa on kaksi pääosiota ja kaksi muuta osiota. Pääosiot ovat, koulutusvideoiden rakennus paloista (Compiling video) ja välispottien luominen kuvatiedostoista (Intermissiontool). Osiot eroteltiin välilehtien alle. HTML-sivun rakenne pyrittiin jakamaan isoihin diveihin joista jokainen käsitti yhden välilehden. Divit ovat HTML-koodin peruselementtejä.

Iso osa sivun HTML:stä generoidaan automaattisesti Knockoutin databindauksilla. Kaikenlaiset listaukset, joista sovelluksen UI suurimmaksi osaksi muodostuu, on kaikki generoitu foreach-bindauksilla, JS:n tallennettujen taulukoiden pohjalta. Bindausten lisäksi knockout mahdollistaa templatejen käytön. Templatet ovat HTML-koodipohjia, joita voi käyttää tarpeen mukaan eri tilanteissa, näistä vain vaihdetaan käytetty data. Esimerkkinä template, jota käytetään tiedostolistausten rakentamiseen. Templateen tarvitsee vain syöttää näytettävän kansion sisältö taulukkona. Tiedostolistauksia sovelluksessa on monia, ja niitä käytetään eri osioissa. Esimerkiksi tallennuskansion haussa ja tiedostojen etsinnässä. Tiedostolistaukset voi tarvittaessa säätää näyttämään vain kansiot.

```
<script id="browserTemplate" type="text/html">
  <strong class="pathRow" data-bind="text: path"></strong>
  <div class="fileTable">
    <div data-bind="foreach: files">
      <div data-bind="attr:{class: browserStyle}">
        <div style="width:45px; float:left; padding-left:5px; padding:2px">
          <img width=32px height=32px data-bind="attr:{src: icon}, click:
          $parent.openFile"/>
        </div>
        <div style="float:left; padding:5px"
             data-bind="click: $parent.openFile">
          <span data-bind="text: filename"></span>
        </div>
        <div style="float:right; padding:5px; padding-right:18px" data-
          bind="if: deletable">
          <img src="images/Symbols/delete.png" data-bind="click:
          $root.deleteFile"/>
        </div>
      </div>
```

```
        </div>
    </div>
</script>
```

4.3 JavaScript-luokat

Sovellus sisältää suuren määrän JavaScript-tiedostoja, joista osa on sovelluksessa käytettyjen frameworkien kirjastoja ja suurin osa sovelluksen osia. JavaScript-kehityksessä tiedostojen määrä on yleensä hyvä pitää pienenä, jotta sivusto latautuu nopeasti.

Tämä ei kuitenkaan ole ongelma, sillä valmis sovellus voidaan ajaa jonkin minifiointin-työkalun lävitse, kuten jscompress.com. Tällainen työkalu pakkaa kaiken koodin yhteen JavaScript-tiedostoon ja lyhentää koodia itseään muun muassa lyhentämällä kaikkien muuttujien nimet mahdollisimman pieniksi. Tällainen 'minifioitu' koodi ei ole kovinkaan luettavaa, mutta ohjelmasta säilytetään alkuperäinen kehitysversio muokkauksia varten.

Koodin rakenteessa pyrittiin seuraamaan Knockout.js:n mukaista MVVM-mallia. Sovelluksessa pyrittiin jakamaan koodin niin, että jokaisesta sovelluksen osiosta tehtäisiin oma View Model, HTML-sivun ollessa View. Näin sovelluksen rakenne olisi looginen: Jokainen välilehti olisi oma Viewinsä, jota päivittää välilehden oma View Model, joka on omassa JS-tiedostossaan.

Esimerkkinä alkua videon rakentamisen View Modelista:

```
function fileListView() {
    fileList = this;
    fileList.videoName = ko.observable("");
    fileList.videos = ko.observableArray(loadAllVideos());
```

Sovelluksen koodin ongelmana on ettei MVVM-mallia ole seurattu loppuun asti. Koodiprojektista puuttua selkeät Modelit, joihin eri View Modelien välissä liikuteltava data voitaisiin tallentaa. Nyt muutamat osat koodista ovat sekavia, koska tieto kulkee View Modelien välissä ilman View Modelit yhdistävää logiikkaa.

4.4 Tiedostojen lähetys palvelimelle ja selaaminen web-käyttöliittymässä.

Käyttäjien täytyy voida lisätä palvelimelle tiedostoja sovelluksen käyttöön. Tämä on mahdollista web-sovelluksen kautta kahdella tavalla, joko raahaamalla tiedoston listauksen päälle, tai

käyttämällä sovelluksen ”file upload”-työkaluja. Nämä toteutettiin käyttäen JavaScriptiä ja PHP:tä, jotka toimivat yhdessä Ajaxilla. Suurin osa sovelluksen koodista on kirjoitettu työn aikana. Tästä poikkeus on tiedostojen lähetys, joka toteutettiin valmiilla vapaasti käytettävällä koodilla.

Palvelimella oleva tiedostorakenne on pyritty pitämään yksinkertaisena. Kaikki sovelluksen kautta lähetettävät tiedostot menevät uploads-kansion alle. Sovelluksen tiedostoselain on rajoitettu niin, ettei tästä kansiossa pääse enää yläkansioihin. Uploads kansion alla on oletuskansiot valmiille videokoosteille, sekä siirtymävideoille ja siirtymävideoihin tarvittaville kuvatiedostoille.

Kattavat tiedostorakenteen muokkausmahdollisuudet eivät olleet tarpeellisia tämän sovelluksen suhteen. Web-sovelluksen kautta voi toistaiseksi vain lisätä ja poistaa tiedostoja. Esimerkiksi tiedostojen siirtäminen kansioiden välillä tai kansioiden luominen ei ole mahdollista. Nämä on tarkoitus tehdä erillään sovelluksesta, joko suoraan palvelinkoneella tai ottamalla palvelimeen yhteyden jollakin sopivalla yhteysohjelmalla. Sovelluksen tiedostoselain on suunniteltu lähinnä vain tiedostojen selausmielessä, ei niinkään muokkausmahdollisuuksia ajatellen.

Tiedostoselaimen näkyvä osa on HTML-template, johon databindataan halutun kansion sisältö. Tämä luo suoraan HTML-muotoisen esityksen halutun kansion sisällöstä. Kansion sisältö saadaan Ajax-kutsulla PHP:stä, joka lukee palvelimelta kansion sisällön ja luo siitä XML-dokumentin. Kansion sisältä luodaan käyttäen PHP:n glob-komentoa. Glob on komento joka etsii kaikki annettua parametriä vastaavat tiedostot kansiosta, jossa PHP-tiedosto on. Jos halutaan etsiä jostain muusta kansiosta, täytyy glob-komennolla antaa parametrinä polku kansioon.

Koodi jolla glob-komento ajetaan:

```
$files = glob("../" . $directory . "*");
```

Eli jos halutaan etsiä vaikka kaikki tiedostot kansiosta /uploads/images, annetaan parametriksi "/uploads/images/*"

4.5 Videokoosteiden rakentaminen

Videokoosteiden rakentaminen on sovelluksen päätehtävä. Joten tämä osio pyrittiin tekemään mahdollisimman vaivattomaksi. (ks. kuvio 1) Videon luominen voidaan aloittaa, joko luomalla tyhjä video tai avaamalla jokin entinen video pohjaksi ja tallentamalla se uudella nimellä.

Tämän jälkeen videoon voidaan lisätä osia valitsemalla niitä tiedostolistauksesta. Kaikki videot joita tiedostolistauksesta klikkaa, menevät osaksi videokoostetta. Tiedostoja voi poistaa klikkaamalla listan roskakori-kuvaketta. Videoita pystyy myös järjestämään uudestaan yksinkertaisesti raahaamalla niitä eri paikkoihin listassa.

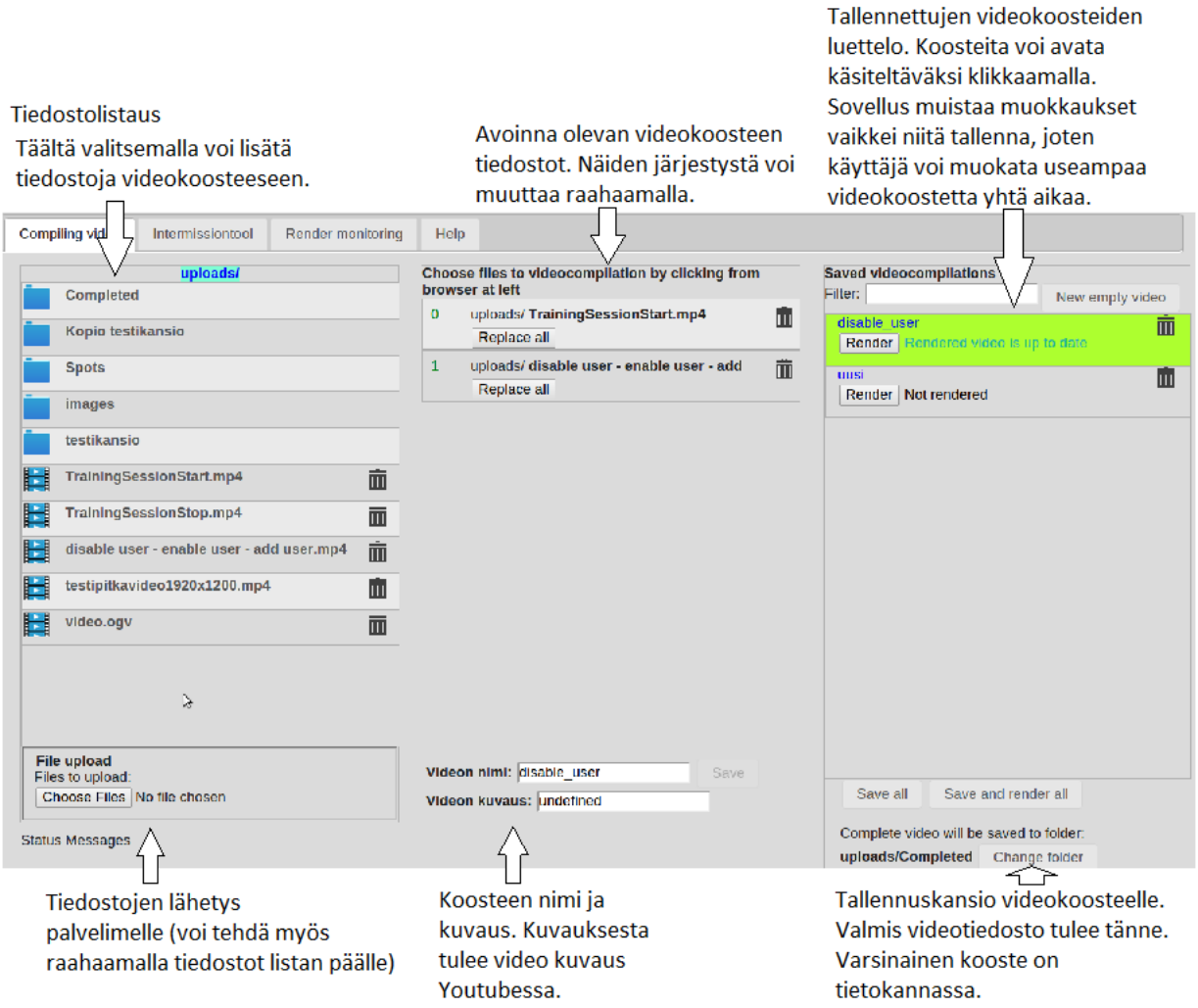
Kun videokoosteessa on kaikki tarpeelliset videot oikeassa järjestyksessä, videokoosteeseen täytyy enää lisätä sopiva kuvaus. Tämä kuvaus on teksti, jonka sovellus laittaa videolla selitykseksi YouTubessa. Sovellus ei toistaiseksi tue erillisiä otsikoita, vaan videon nimeksi YouTubessa tulee videon tallennusnimi. Kun kuvaus on sopiva, video voidaan tallentaa, tallennettu video laitetaan talteen MySQL-tietokantaan. Kooste itsessään ei mene tietokantaan, sinne tallennetaan vain tieto koosteen sisältämistä videoista ja niiden järjestys.

Teknisesti tämä kaikki on toteutettu sovelluksen JavaScript-koodissa selaimella. Videotiedostoja voi järjestellä ja hakea tiedostolistauksesta puhtaasti web-käyttöliittymällä, ilman että tarvitsee kutsua palvelimelta mitään lisää. Kaikki tietokantaan tallennetut videokoosteet ladataan yhteen luetteloon sovelluksen käynnistyessä. Tämän ei arvioitu olevan liian kuormittavaa, sillä videokoosteita parhaimmassakin tapauksessa tulee mahdollisesti joistakin sadoista tuhanteen. Ennemminkin ongelmaksi muodostui tallennettujen videokoosteiden lista, joka helposti leviää valtavan pitkäksi ja tekee halutun videokoosteen löytämisestä hyvin vaikeaa. Ratkaisuksi tähän kehitettiin filtteri, jolla voi hakea videokoosteita jotka sisältävät tietyn sanan tai merkin. Filtterin luominen oli hyvin yksinkertaista:

```
if (video.videoName().indexOf(filter) > -1) {
```

Koodi tarkistaa sisältääkö videokoosteen nimi halutun sanan yhteen tai useampaan kertaan.

Alla esimerkki knockout-sortablen käytöstä videokoosteen tiedostojen luetteloinnissa. Knockoutin observablearrayn yhdistäminen sovellukseen manuaalisesti muokattavana listana on hyvin yksinkertaista ja vaatii vain yhden bindauksen.



KUVIO 4. Videokoosteiden rakentamisen käyttöliittymä.

4.6 Sovelluksen tietokannan toiminta

Sovellus käyttää MySQL-pohjaista tietokantaa. Sovelluksen tietokanta on hyvin yksinkertainen ja sisältää vain kolme taulua. Monimutkainen ja moneen taipuva tietokanta ei ollut tämän työn suuria prioriteetteja. Tietokannassa on vain kaikki välttämätön videokoosteiden tallennukseen.

Tietokannan toimintaperiaate rakentuu yksittäisten videotiedostojen ympärille, enemmän kuin videokoosteiden. Tietokannassa pyritään välttämään samojen videotiedostojen tallentamista moneen kertaan. Joka kerta kun käyttäjä tallentaa uuden videokoosteen, kaikki videokoosteessa olevat videotiedostot yritetään etsiä tietokannasta. Tietokantaan on

tallennettu myös tiedoston timestamp, eli numeerinen arvo siitä, milloin tiedostoa on viimeksi muokattu. Vertaamalla tietokantaan tallennettua timestampia ja uuden videotiedoston timestampia saadaan varmuus siitä onko kyseessä sama tiedosto. Mikäli samaa videotiedostoa on jo käytetty aikasemmin (timestamp, videon nimi ja sijainti vastaavat), tässä uudessa videokoosteessa käytetään näitä jo tietokannasta löytyviä videotiedostoja, eikä luoda uusia. Videotiedostojen pakottaminen samaksi on myös välttämätöntä korvatessa samaa videotiedostoa useammasta videokoosteesta yhtä aikaa. Näin vältytään fyysisten tiedostojen vertailulta, voidaan vain korvata yksi videotiedosto tietokannasta.

Tietokantaan sovellus on yhteydessä PHP:n välityksellä. PHP-tiedostoja kutsutaan Ajax-kutsuilla web-sivulta käsin. Tietokannasta haetaan JavaScriptiin tietoa XML-muodossa. Ajaxilla kutsutaan PHP-sivua, joka rakentaa XML-dokumentin halutusta tiedosta. Tätä XML-dokumenttiä voi sen jälkeen helposti lukea JavaScriptissä ja muokata se sopivaan muotoon, jotta se voidaan esittää HTML-rakenteena databindauksen avulla. Alla esimerkki tallennettujen videokoosteiden listauksesta XML-muodossa.

```
<XML>
<Video ID="6" Videoname="adding_stuff" Rendered="" Summary="undefined">
                <File ID="1" Filename="TrainingSessionStart.mp4" Path="uploads/"
                Modified="1360855603" NotFound="0"/>
                <File ID="10" Filename="add_stuff.mp4" Path="uploads/"
                Modified="1375701933" NotFound="0"/>
                <File ID="8" Filename="TrainingSessionStop.mp4" Path="uploads/"
                Modified="1360855617" NotFound="0"/>
</Video>
<Video ID="7" Videoname="adding_stuff_part2" Rendered="" Summary="undefined">
                <File ID="1" Filename="TrainingSessionStart.mp4" Path="uploads/"
                Modified="1360855603" NotFound="0"/>
                <File ID="10" Filename="add_stuff.mp4" Path="uploads/"
                Modified="1375701933" NotFound="0"/>
                <File ID="11" Filename="add_more_stuff.mp4" Path="uploads/"
                Modified="1375701933" NotFound="1"/>
                <File ID="8" Filename="TrainingSessionStop.mp4" Path="uploads/"
                Modified="1360855617" NotFound="0"/>
</Video>
<Video ID="5" Videoname="projektin_luonti" Rendered="" Summary="undefined">
                <File ID="1" Filename="TrainingSessionStart.mp4" Path="uploads/"
                Modified="1360855603" NotFound="0"/>
                <File ID="9" Filename="create_project.mp4" Path="uploads/"
                Modified="1375701933" NotFound="0"/>
                <File ID="8" Filename="TrainingSessionStop.mp4" Path="uploads/"
                Modified="1360855617" NotFound="0"/>
</Video>
<Video ID="8" Videoname="user_testing" Rendered="" Summary="undefined">
```

```
            <File ID="1" Filename="TrainingSessionStart.mp4" Path="uploads/"
            Modified="1360855603" NotFound="0"/>
            <File ID="12" Filename="test_user.mp4" Path="uploads/"
            Modified="1375701933" NotFound="0"/>
            <File ID="8" Filename="TrainingSessionStop.mp4" Path="uploads/"
            Modified="1360855617" NotFound="0"/>
</Video>
</XML>
```

XML-dokumentista voi nähdä, kuinka sovellus lataa videokoosteet (<Video>) ja niiden sisältämät videotiedostot(<File>). Videotiedostoissa oleva NotFound-arvo kertoo onko fyysistä videotiedostoa löydetty kansiosta esitarkistuksessa.

4.7 Siirtymävideoiden luominen

Siirtymävideoiden luonti on toinen sovelluksen päätehtävistä. Tämä on toteutettu omalla välilehdellään sovelluksen käyttöliittymässä. Siirtymävideoiden luonti on yksinkertaista. Tämän osion käyttöliittymä on rakennettu samalle pohjalle kuin videokoosteiden rakennus. Tiedostolistaus on pidetty paikoillaan ja keskisarake on korvattu käyttöliittymällä, jolla säädetään asetukset halutulle siirtymävideolle. (ks. kuvio 2) Tiedostovalikosta valitaan kuvatiedosto, josta siirtymävideo halutaan tehdä. Sovellus tukee PNG- ja JPG-kuvatiedostomuotoja. Tämän jälkeen säädetään asetukset. Kun asetukset ovat kohdillaan, annetaan siirtymävideolle nimi ja rendataan se haluttuun kansioon. Rendattu siirtymävideo on tämän jälkeen suoraan käyttäjän käytettävissä videokoosteiden rakentamisessa.

Teknisesti siirtymävideoiden rakentaminen onkin ehkä sovelluksen mutkikkain osa. Aloittaessa rendauksen sovellus käynnistää rendauksesta vastaavan PHP:n antamalla sille käyttäjän syöttämät muuttujat. Tämän jälkeen PHP ajaa palvelimella olevan Python-skriptin. Python-skriptille lähetetään samat muuttujat jotka PHP sai web-sovelluksesta. Näihin muuttujiin sisältyy muun muassa kuvatiedoston nimi ja sijainti, joiden perusteella Python-skripti etsii kuvatiedoston palvelimelta ja käynnistää avconv-työkalun, jolla varsinainen rendaus tehdään. Eli siirtymävideoiden luonti on hyvin moniportainen toimenpide joka kulkee seuraavanlaisen polun:

Web-käyttöliittymä (JS) -> Palvelin (PHP) -> Python-skripti -> Palvelimen komentorivi (avconv-komento)

Alla esimerkki PHP-koodista, joka rakentaa python-skriptin käynnistävän komennon:

```
$command = "python spot.py " . $image . " " . $videoname . " " . $path . " ";
$command .= $length . " ";

if ($noScaling) {
        $command .= "noScaling noScaling";
} else {
        $command .= $width . " " . $height;
}

$command .= " Video > templog.txt 2>&1 &";
shell_exec($command);
```

Koodissa rakennetaan komento PHP:lle syötetyistä muuttujista ja suoritetaan python-skripti spot.py PHP-komennolla shell_exec.

Seuraavana taas esimerkki avconv:in käynnistyskomennon rakentamisesta Pythonilla:

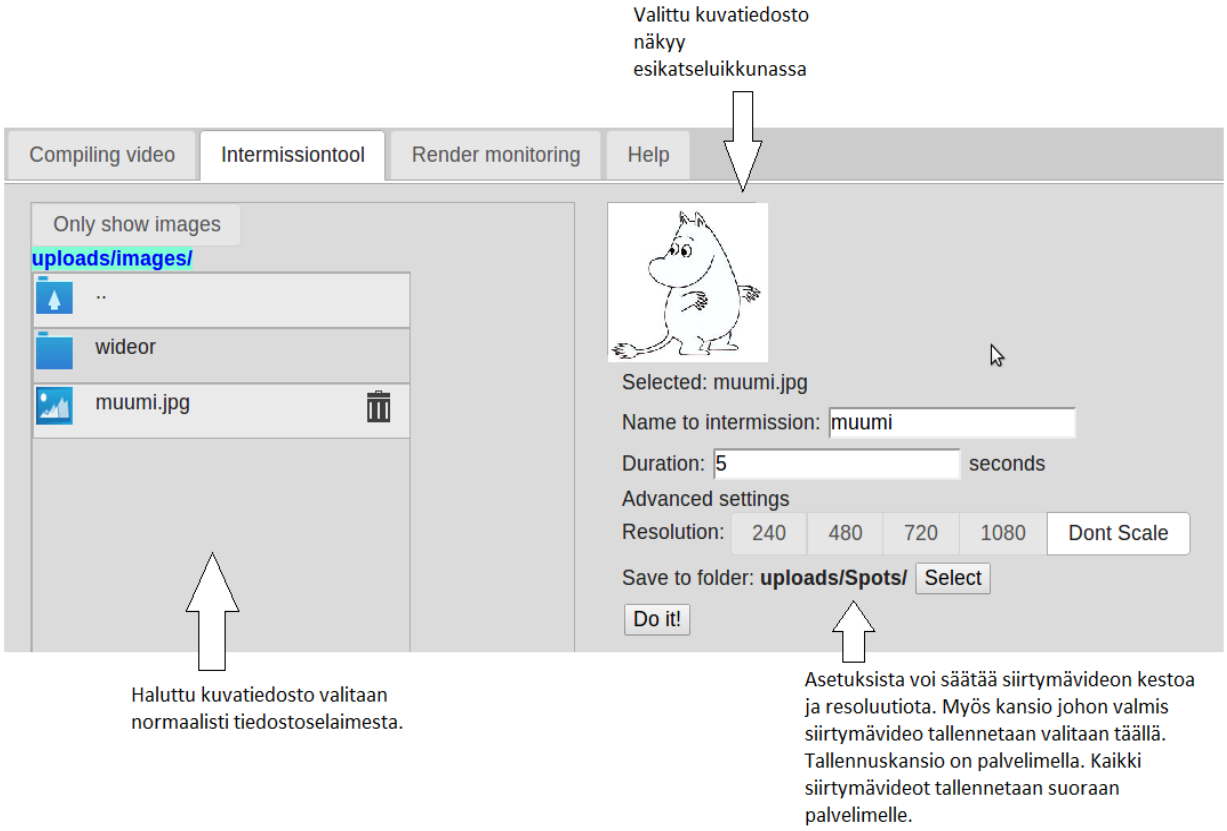
```
command = "avconv -loop 1 -f image2 -i"
command += " ../" + image
command += " -t " + length

if width != "noScaling":
        command += " -s " + width + "x" + height

fullpath = "../" + path + videoname + ".mp4"
command += " " + fullpath

command += " > " + logfile
os.system(command);
```

Kuten koodiesimerkistä näkyy, toimenpide on hyvin samanlainen. Muuttujat syötetään komennolle, joka suorittaa avconvin. Syntaksi on tietenkin hieman erilainen. Toimenpide eroaa PHP:stä siinä, että tässä myös annetaan tiedosto (logfile), johon avconv kirjoittaa prosessin etenemisen. Tätä tiedostoa tarvitaan myöhemmin rendauksen seurannassa.



KUVIO 5. Siirtymävideoiden rakennuksen käyttöliittymä.

4.8 Videokoosteiden lähetys YouTubeen

Viimeinen porras videon rendauksen valmistuttua on julkaista valmis video YouTubessa. Myös tämä toiminto tehdään automaattisesti palvelimella. Videoiden julkaisu tehdään käyttäen GoogleCl-työkalua. Tämä työkalu ajetaan rendauksen valmistuttua sovelluksen Python-skriptissä. Alla esimerkkinä Python-koodi joka käynnistää GoogleCl:n:

```
subprocess.call(["google", "youtube", "post", "--category", "Education", "--summary", summary , "--user", "markustestailua" , fullpath])
```

Komennossa käynnistetään GoogleCL:n "youtube post"-toiminto. Tämä vaatii muuttujina videon polun, kuvauksen videolle sekä käyttäjän, jonka tilillä video julkaistaan. Käyttäjän (tässä tapauksessa testitili "markustestailua". Lopullisessa sovelluksessa käytetään FreeNestin YouTube-tiliä) on täytynyt aiemmin antaa oikeudet palvelinkoneelle julkaista videoita hänen tilillään. YouTube myös vaatii, että jokainen video sisältää kategorian. Oikeudet annetaan tapaan, tietokoneen nimi ja tietokoneen käyttäjän nimi. Tietokoneen käyttäjä on aina www-data, koska tämä on oletuskäyttäjä, jolla palvelimen käynnistämät sovellukset suoritetaan Ubuntu-ympäristössä. Sovellus käyttää aina kategoriaa "education". Tätä ei toistaiseksi voi vaihtaa kuin koodista käsin. GoogleCL:ää ei voinut laittaa tulostamaan toiminnon etenemistä tekstitiedostoon, joten lopullinen lokitiedoston päivitys tehdään erikseen, mikäli lähetys onnistui ongelmitta. Mallina Python-koodi joka lisää lokiin tekstin "Completed".

```
log = open(logfile, 'w')
log.write('Completed')
log.close()
```

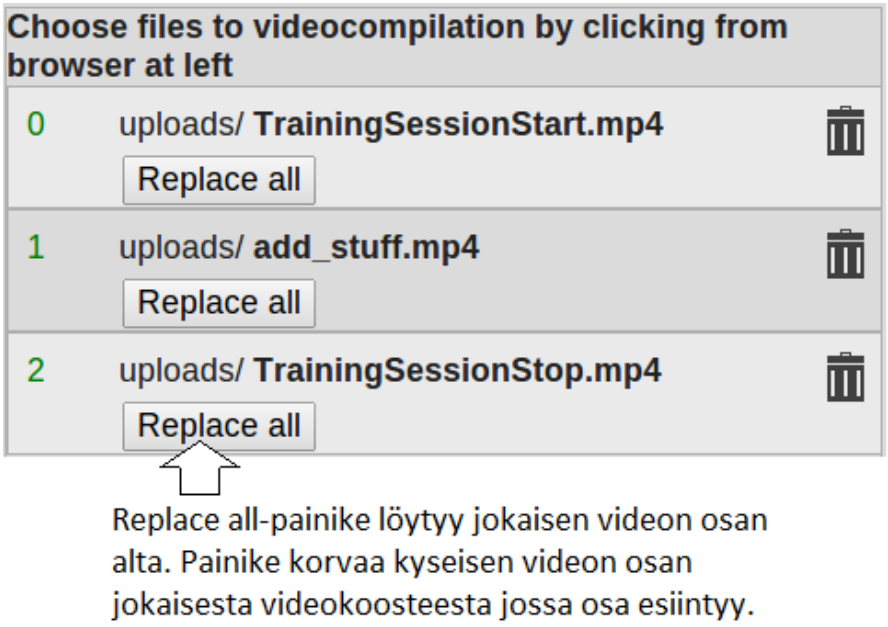
4.9 Muokkausten peilaus muihin videokoosteisiin

Toiminto, joka ehdottomasti haluttiin sovellukseen mukaan, oli mahdollisuus muokata useampia videokoosteita kerralla. Toisinaan koulutusmateriaalia päivittäessä saattaa olla tarpeen vaihtaa jokin osio monesta videosta. Esimerkiksi, jos viime vuonna tehtyihin videoihin halutaan vaihtaa tämän vuoden aloitusosio. Tämä toiminto toteutettiin niin, että valitsemalla jokin videokooste, jossa vaihdettava video esiintyy, se voidaan vaihtaa kaikista videoista kerralla käyttäen sovelluksen "Replace all"-toimintoa (ks. kuvio 3).

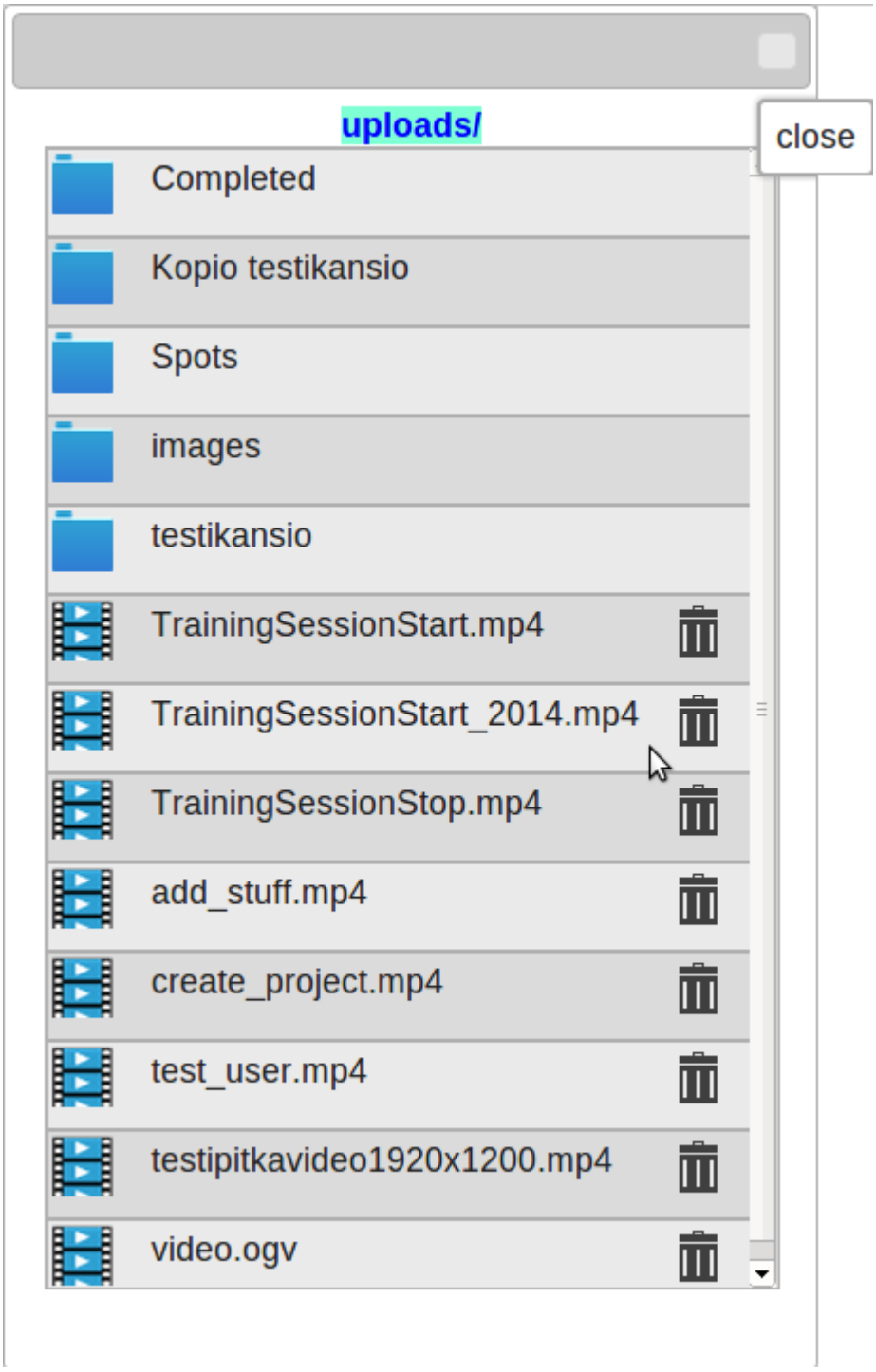
"Replace all" avaa oman pop-up näkymän, jossa vaihtaminen tehdään. Ensimmäisenä eteen tulee pop-up ikkunaan avautuva instanssi tiedostoselaimesta, josta valitaan korvaava video (ks.

kuvio 4). Tämän jälkeen sovellus etsii kaikki videokoosteet, joissa vaihdettava video esiintyy. Nämä videokoosteet listataan seuraavaksi esiin tulevaan ikkunaan (ks. kuvio 5). Tässä vaiheessa on mahdollista käyttäjän valikoida listasta ne videokoosteet, johon vaihdon haluaa ulottuvan.

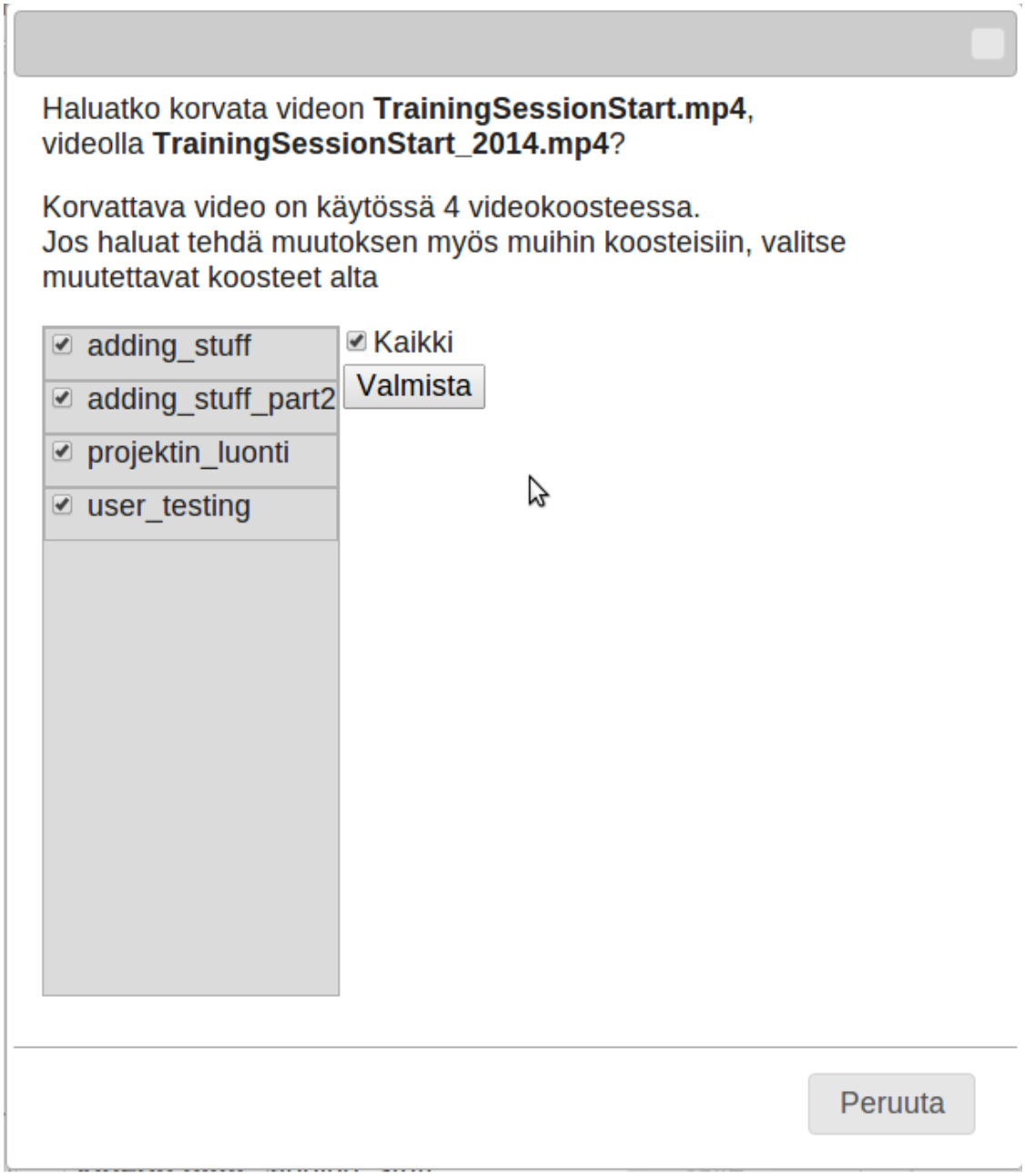
Teknisesti vaihto tapahtuu MySQL-queryllä, joka vaihtaa videon osat tietokantaan tallennettuihin koosteisiin. Vaihto ei siis tapahdu "fyysisesti", eli videot eivät suoraan muutu. Saadakseen YouTubeen uudet päivitetyt rendaukset videoista, käyttäjän on rendattava muuttuneet videot uudestaan. Tämä voidaan tehdä nopeasti painamalla "Render all"-painiketta tallennettujen videokoosteiden listassa (ks. kuvio 6). YouTuben toimintatavoista johtuen videoita ei voi suoraan korvata uusilla, joten videolinkkien päivittäminen materiaaliin, tai missä videoita sitten käytetäänkään, jää edelleen käyttäjän vastuulle. Sovellus vain lähettää Youtubeen uudet versiot videoista, joille Youtube luo uudet osoitteet.



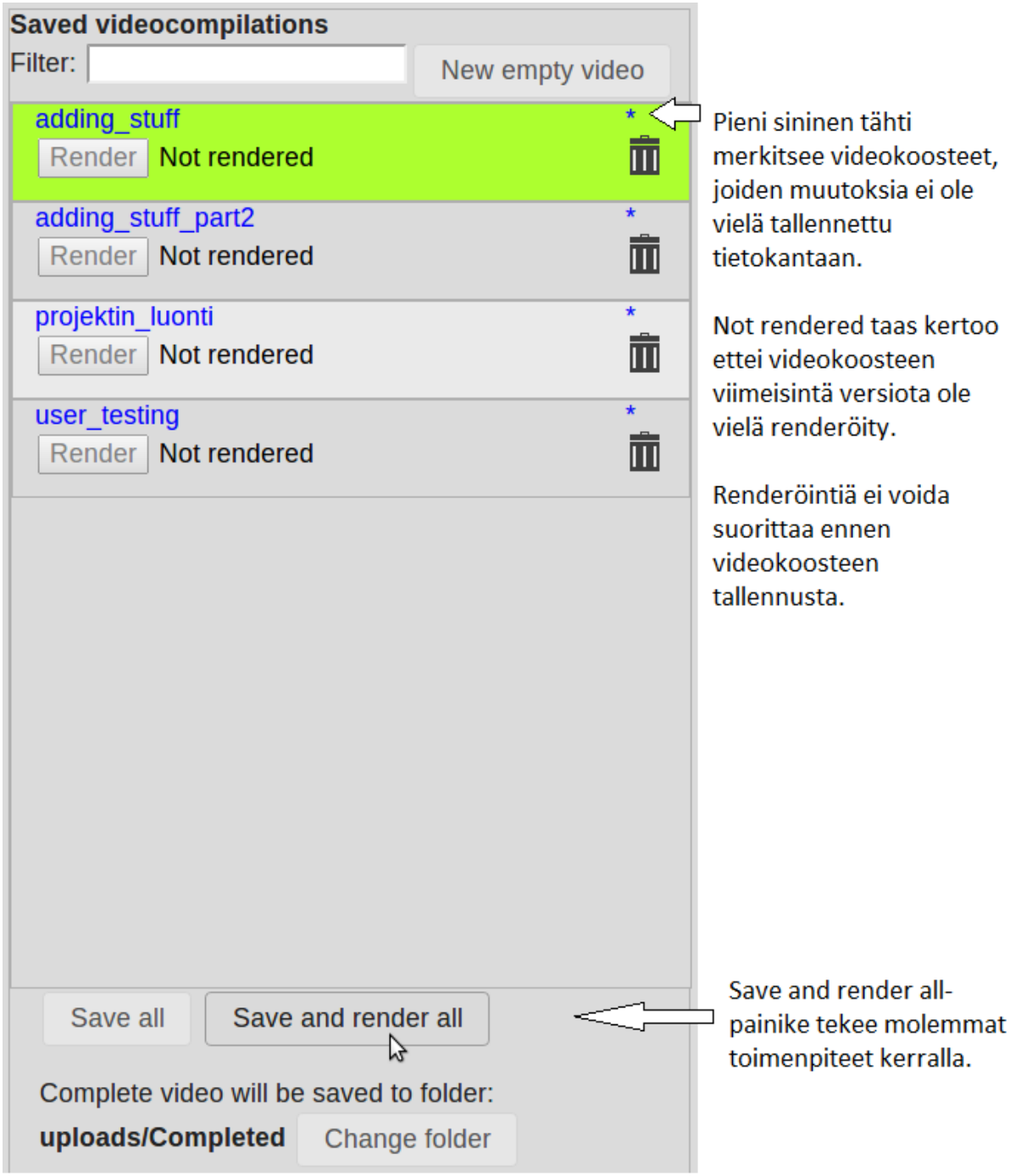
KUVIO 6. Videoiden korvaaminen aloitetaan tästä.



KUVIO 7. Korvaavan videotiedoston valitseminen tiedostoselaimesta.



KUVIO 8. Muutosten peilaaminen useampaan videokoosteeseen.



KUVIO 9. Useiden videokoosteiden rendaaminen yhtä aikaa.

4.10 Rendauksen ja muun käsittelyn seuraaminen

Rendauksen seuranta on sovelluksen kolmas osio. Tämä on välilehti, josta selviää mitä palvelin tällä hetkellä tekee. Aina kun käyttäjä käynnistää jonkin palvelimella tehtävän toimenpiteen, se lisätään tässä osiossa näkyvään listaan. Toistaiseksi tässä osiossa näkyy vain missä vaiheessa toimenpide on menossa. Esimerkiksi videokoosteen rendauksessa vaiheet ovat: valmistelu, rendaus ja lähetys YouTubeen, sekä "Complete" kun nämä vaiheet on saatu toteutettua (ks. kuvio 7).

Teknisesti tämä osio on toteutettu seuraamalla lokitiedostoja, joita Python-skriptit päivittävät sitä mukaa kun tapahtumat etenevät palvelimella. Käyttäjän käynnistäessä mikä tahansa palvelinpuolen toimenpide web-käyttöliittymässä, luodaan JavaScriptiin kuuntelija, joka lukee tiettyä tähän toimenpiteeseen liittyvää lokitiedostoa. Nämä kuuntelijat päivittävät toimenpiteen tilan sekunnin välein lukemalla lokitiedoston aina uudestaan. Sekuntin päivitysnopeuteen päädyttiin, sillä tarkemman seurauksen ei katsottu olevan tarpeellista ja suurempi nopeus aiheuttaisi tarpeetonta kuormaa palvelimelle. Mikäli sovellukseen tullaan tekemään edistymispalkit, pelkän vaiheen sijaan, voidaan palkit "huijata" päivittymään nopeammin ilman että käyttäjä huomaa mitään. Tällöin sovellus päivittää palkkia kokoajan, mutta vain pienen osan kuuntelijan ilmoittamasta etenemisestä kerrallaan.

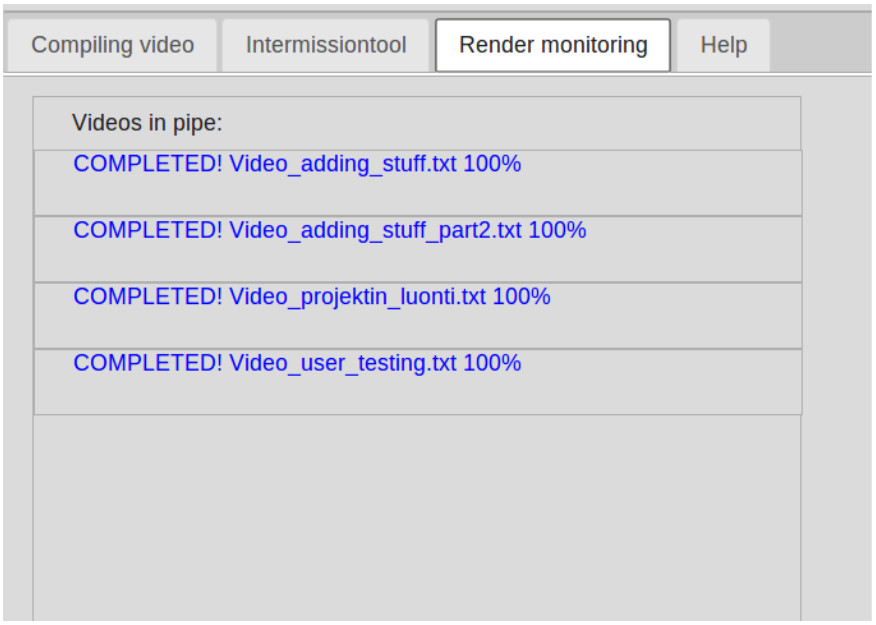
Esimerkki lokitiedoston käsittelystä sovelluksessa, mikäli kyseessä on videokoosteen rendaaminen: (Lokitiedosto on luettu Ajax-kutsulla)

```
case "Video":
        if (result.indexOf("ISO File Writing") > -1)
                item.updateProgress(1);
        }

        if (result.indexOf("Uploading") > -1){
                item.updateProgress("Uploading");
        }

        if (result.indexOf("Completed") > -1) {
                item.updateProgress("Completed");
        }
        item.updateProgress(0);
        break;
```

Käsittelijästä voi nähdä, että sovellus lukee missä vaiheessa operaatio on menossa. Käsittelijästä näkee myös sen että varsinaiset edistymispalkit puuttuvat, ainoastaan meneillään oleva vaihe luetaan.



KUVIO 10. Rendauksen seurantavälilehti.

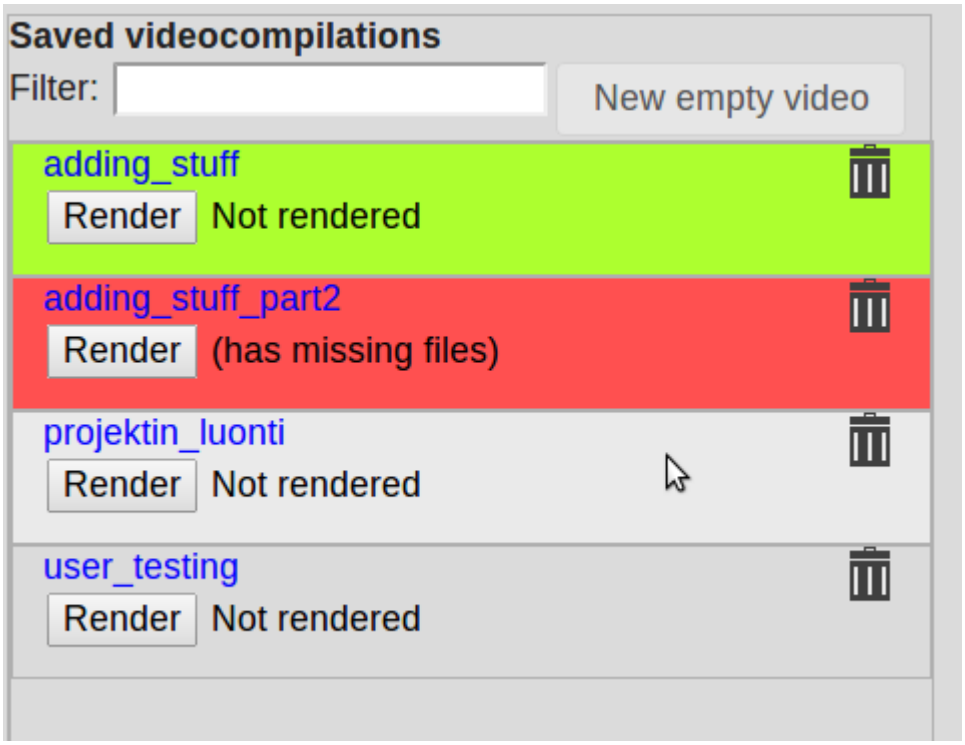
4.11 Tiedostojen muutosten havaitseminen ja käsittely

Videotiedostoja ei itsessään tallenneta tietokantaan, vaan ne ovat suoraan esillä ja muokattavissa palvelimella. Tästä johtuen on mahdollista, että käyttäjä jolla on suora pääsy palvelimelle, voi esimerkiksi poistaa videotiedostoja, jotka ovat käytössä joissakin videokoosteissa. Tällaisia tilanteita varten sovellukseen on tehty toiminto, joka aina sovelluksen web-käyttöliittymän avatessa tarkastaa kaikkien videokoosteiden eheyden. Kyseessä on PHP-tiedosto, joka käy lävitse kaikki tallennetut videokoosteet tietokannasta ja tarkistaa, että tiedostot joihin videokoosteissa viitataan ovat edelleen tallella, eivätkä ne ole muuttuneet.

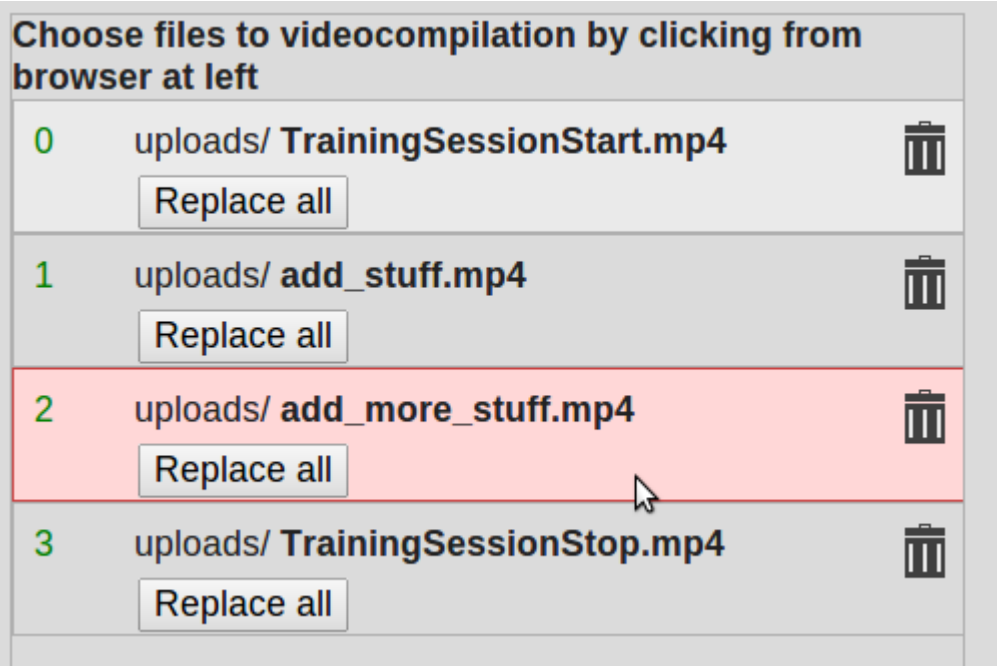
Mikäli puuttuvia tiedostoja tai muutoksia havaitaan, lisätään tietokantaan merkintä virheestä videokoosteessa. Tällaiset virheelliset videokoosteet näkyvät listassa punaisina (ks. kuvio 8). Avaamalla tällaisen videokoosteen, käyttäjä saa myös tarkempaa tietoa siitä, mikä koosteessa

on tarkalleen vialla. Videolistauksessa näkyvät punaisina ne videotiedostot, joita sovellus ei löytänyt (ks. kuvio 9).

Tässä vaiheessa sovellus mahdollistaa videokoosteiden korjaamisen manuaalisesti. Mikäli puuttuvat tiedostot on esimerkiksi siirretty toiseen paikkaan, voi käyttäjä käyttää "Replace all"-toimintoa, etsiä tiedostot käsin ja päivittää tiedostojen sijainnit helposti kaikkiin videokoosteisiin. Tai mikäli tiedostot ovat kadonneet kokonaan, voi käyttäjä lähettää ne palvelimelle takaisin. "Replace all"-toiminnon pitäisi tehdä korjaukset helpoksi, sillä se korjaa videotiedoston suoraan kaikista videokoosteissa missä tämä esiintyy. Videoiden uudelleen rendaaminen ei ole tarpeellista korjaamisen jälkeen, sillä jo rendatut videot pysyvät joka tapauksessa kunnossa YouTubessa. Sovelluksella ei ole vaikutusta valmiisiin videoihin.



KUVIO 11. Tallennettujen videokoosteiden luettelossa on videokooste, jonka tiedostoista osa on kadonnut.



KUVIO 12. Viallinen videokooste avattuna. Sovellus ei ole löytänyt videotiedostoa add_more_stuff.mp4 ja näyttää sen puuttuvan.

5. JATKOKEHITYSSUUNNITELMAT

Sovelluksessa on vielä yleisesti paljon viimeisteltävää, jotta se sopisi normaaliin käyttöön. Monet hyvin välttämättömät asiat eivät toimi kunnolla, kuten resoluution määritys siirtymävideolle. Jos siirtymävideolle yrittää antaa liian suuren resoluution sovellus kaatuu. Tämä on hyvin ongelmallista sillä videokoosteiden rakentamisessa automaattinen skaalaus ei myöskään toimi. Videoiden täytyisi olla samalla resoluutiolla, jotta videokooste voidaan rakentaa. Sovelluksessa on myös paljon pienempiä, jotkin nappulat eivät muutu painettaviksi silloin kuin niiden pitäisi, esimerkiksi jos videokoosteeseen on tehnyt korvauksia ja tallentanut sen, "replace all"-ei välttämättä tule takaisin painettavaksi.

Kuten työn kuvioistakin näkee, myös yleistä viimeistelyä tarvittaisiin vielä paljon. Lopullinen sovellus olisi tarkoitus olla kokonaan englannin kielinen, mutta sovelluksessa on vielä paljon kääntämättömiä kohtia. Sovelluksen layout on myös hyvin keskeneräinen, monia yksityiskohtia puuttuu, kuten sovelluksen otsikkoalueen koristelu. Myöskään välilehtien esitys ei ole layout-suunnitelman mukainen. Lisäksi joitain osioita, kuten "replace all"-ikkuna, ei ole vielä suunniteltu ollenkaan.

Sovelluksen käyttöliittymä kaipaisi myös hiomista. Käyttöliittymän flow, eli käyttäjän intuitiivinen eteneminen sovelluksen läpi ei ole kohdallaan. Esimerkiksi videokoosteiden rakennuksessa täytyy hyppiä eripuolille sovellusta. Uusi video luodaan ensin oikeasta laidasta, jonka jälkeen siirrytään vasemmalle lisäämään osia ja tämän jälkeen keskelle. Videokoosteiden rakentaminen on myös osittain epäyhtenäinen, videoiden järjestystä muutetaan raahaamalla, kun taas uusia videoita lisätään klikkaamalla tiedostolistauksesta. Sovellusta esitellessä jotkut ovat yrittäneet raahata videoita tiedostolistauksesta.

Myös sovelluksen ylläpitopuoli vaatisi vielä paljon kehitystä. Toistaiseksi sovelluksen asentaminen uuteen koneeseen on monimutkaista. Kaikki sovelluksen vaatimat laajennukset ja työkalut täytyy asentaa ensin manuaalisesti (muun muassa GoogleCl, avconv, MP4Box, sekä näiden vaatimat oheiskirjastot). Myöskin YouTube-tilin, johon sovellus on kiinnitetty, vaihtaminen on hyvin monimutkainen prosessi. Tilin tiedot täytyy vaihtaa Python-koodista ja käyttöoikeuksien antaminen palvelinkoneelle tässä uudessa tilissä on hankalaa. Käytännössä täytyy tehdä testilähetyksiä manuaalisesti ja tämän jälkeen käydä selaimella sallimassa pyynnöt. Tämä on vieläpä ongelmallista, sillä käyttöikeuksien vahvistusikkuna, avautuu selaimeen vain, mikäli GoogleCL on ajettu manuaalisesti sillä koneen käyttäjällä, jolla lähetyksiä tullaan tekemään. Palvelimen käynnistämien sovellusten käyttäjä on aina www-data, jota ei varsinaisesti kuuluisi voida käyttää Ubuntussa manuaalisesti. Tämän käyttäjätiliongelman selvittäminen kuuluu myös sovelluksen jatkokehitykseen.

6. YHTEENVETO JA JOHTOPÄÄTÖKSET

Lähdin rakentamaan sovellusta knockout.js:llä koska se oli hyvin uusi framework ja sen opettelu vaikutti erittäin hyvältä idealta. Jälkikäteen olen tyytyväinen valintaani. Knockout helpotti huomattavasti käyttöliittymän rakentamista. Erilaisten toimintojen ja painikkeiden lisääminen oli erittäin vaivatonta koko toteutusvaiheen ajan. Ongelmia koodissa tuli siitä, etten aloittaessa vielä hahmottanut kunnolla knockoutin Model-View-View Model-filosofiaa. Tästä johtuen koodista ei tullut kovinkaan selkeää. Koodissa täytyisi olla selkeitä Model-luokkia, joihin sovelluksen data voitaisiin säilöä, nyt käytännössä kaikki data on View Modeleissa. View Modelien kuuluisa vain lukea dataa Modeleista ja muokata se esityskelpoiseksi.

Lähdin myös tämän työn myötä opettelemaan PHPStormin käyttöä sovelluskehityksessä, tähän olen myöskin tyytyväinen. Opinnäytetyössä täytyi myös tutustua Pythoniin, jota tarvittiin palvelinpuolen toiminnoissa. PHP ei pysty tekemään kaikkea palvelinpuolen toimintaa, kuten

erillisten komentorivisovellusten ajamista kovin hyvin. Pythonin opiskelu sujui hyvin kivuttomasti, sillä Python on suunniteltukin mahdollisimman helpoksi oppia.

Merkittävimmät ongelmat liittyivät ubuntu-pohjalla pyörivän apache2-palvelimen toiminnan opiskeluun. Eri käyttäjien ja käyttäjäoikeuksien tutkiminen ja säätäminen kohdalleen kulutti huomattavasti aikaa. Esimerkiksi palvelimen skriptit ajetaan tietyllä käyttäjällä (www-data). Tästä johtuen PHP- ja Python-skriptit, sekä ennen kaikkea eri kolmansien osapuolien työkalut (avconv, GoogleCl) oli hyvin työlästä saada suorittumaan oikein. Mutta lopulta näistä ongelmista suurimmaksi osin selvittiin.

Videoiden rendaamiseen, kuten videotiedostojen skaalaamiseen samankokoisiksi liittyviä ongelmia sovellukseen jäi. En päässyt tarpeeksi pitkälle MP4Boxin ja avconvin tutkimisessa, jotta olisin selvittänyt, mikä ongelmat aiheuttaa.

Loppujen lopuksi sovellus suunniteltiin ehkä liian laajaksi opinnäytetyön työmäärään nähden. Tästä johtuen työ jäi viimeistelemättömäksi. Olen kuitenkin tyytyväinen siihen määrään asioita, jotka sain koodausprosessin aikana valmiiksi.

Tutkittuani syvemmin jo olemassa olevia web-pohjaisia videoeditoreita, en ole enää varma oliko oman sovelluksen tekeminen erittäin hyödyllistä. YouTuben editori sisältää suurimman osan tähän sovellukseen halutuista toiminnoista ja on varmatoimisempi, ainakin verrattuna viimeistelemättömäksi jääneeseen sovellukseen. Käyttöliittymä on mielestäni myös parempi, vastaavanlainen toteutus oli myös tämän työn suunnitelmissa, mutta siihen ei ollut resursseja opinnäytetyön mittakaavassa.

Muista työssä käytetyistä ohjelmista haluan vielä erikseen mainita Ubuntuun valmiiksi asennetun tekstinkäsittelyohjelman LibreOffice, joka on jos mahdollista vielä Word 2007:kin kehnompi tekstinkäsittelyohjelma.

LÄHTEET

About MySQL. Viitattu 28.7.2013
http://www.mysql.com/about/

About Python. Viitattu 22.7.2013
http://www.python.org/about/

Davis, I. 2008. What Are The Benefits of MVC. Viitattu 12.7.2013
http://blog.iandavis.com/2008/12/09/what-are-the-benefits-of-mvc/

Event Driven Programming. 2013. Viitattu 20.7.2013
http://c2.com/cgi/wiki?EventDrivenProgramming

FAQ on MySQL vs. NuSphere Dispute. 2001. Viitattu 26.7.2013
http://web.archive.org/web/20010717185237/http://mysql.com/news/article-75.html

FreeNest-Product Platform. 2013. Viitattu 14.7.2013
http://freenest.org/about

Garrett, J. 2005. Ajax: A New Approach to Web Applications. Viitattu 21.7.2013
http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

General Information. 2013. PHP.net www-sivut. Viitattu 23.7.2013
http://fi1.php.net/manual/en/faq.general.php

General Python FAQ. Viitattu 22.7.2013
http://docs.python.org/2/faq/general.html#id5

Heilmann, C. 2009. What can you do with JavaScript. Viitattu 23.7.2013
http://dev.opera.com/articles/view/javascript-uses/

History of PHP. 2013. Viitattu 23.7.2013
http://www.php.net/manual/en/history.php.php

Hopkins, C. 2013. The MVC-pattern and PHP. Viitattu 4.9.2013
http://www.sitepoint.com/the-mvc-pattern-and-php-1/

Implementing the MVVM pattern. 2013. Viitattu 21.7.2013
http://msdn.microsoft.com/en-us/library/gg405484(v=pandp.40).aspx

JavaScript introduction. W3Schools www-sivut. Viitattu 23.7.2013
http://www.w3schools.com/js/js_intro.asp

JQuery Foundation. 2013. Viitattu 20.7.2013
https://jquery.org/

Knockout: Introduction. Viitattu 12.7.2013
http://knockoutjs.com/documentation/introduction.html

Knockout: Observables. Viitattu 21.7.2013
http://knockoutjs.com/documentation/observables.html

MikaBug. 2007. Ohjelmointiputka: Opas-arkisto. Viitattu 23.7.2013
http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=js_01

Mikä on MOOC? Viitattu 11.9.2013
http://mooc.cs.helsinki.fi/content/mik%C3%A4-mooc

MP4Box | GPAC. Viitattu 26.8.2013
http://gpac.wp.mines-telecom.fr/mp4box/

Projektit. JAMKin www-sivut. Viitattu 14.7.2013
http://www.jamk.fi/projektit/1233

Tietoja Lisensseistä, Creative Commons, Viitattu 29.9.2013
http://creativecommons.org/licenses/

Vairimaa, R. 2013. Yliopistolainen 2/2013. JOUSTAVA MOOC.
http://palvelut.unigrafia.fi/yliopistolainen_2_2013/

What can PHP do. 2013. Viitattu 23.7.2013
http://www.php.net/manual/en/intro-whatcando.php

Why use python? 1999. Viitattu 1.9.2013
http://www.python.org/doc/essays/ppt/acm-ws/sld012.htm